AÑO II - NUM. 55

MICROHOBBY

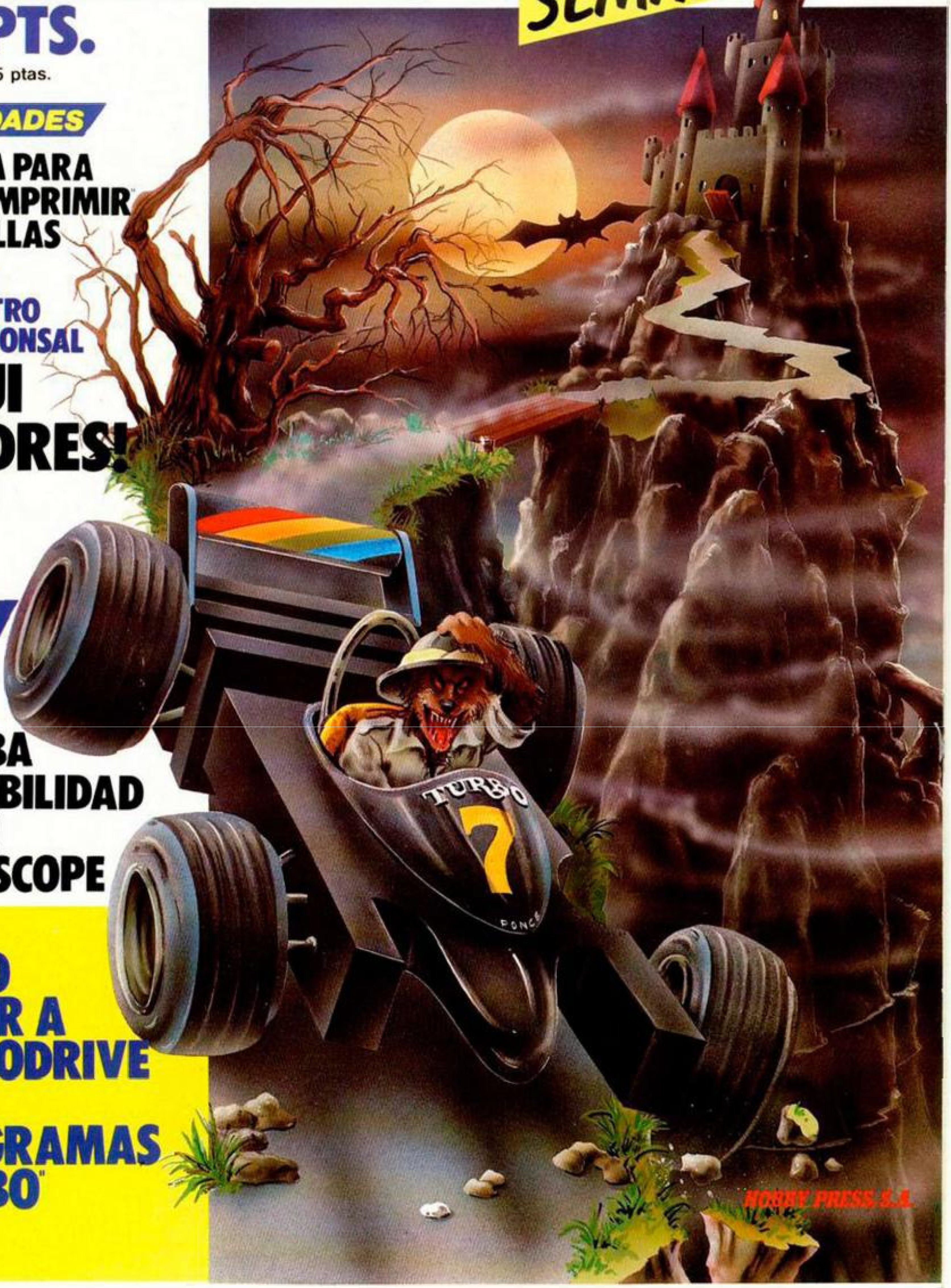
SEMANAL

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES SINCLAIR

AÑO II - N.º 55

125 PTS.

Canarias 135 ptas.



UTILIDADES

RUTINA PARA "DESCOMPRIMIR" PANTALLAS

DE NUESTRO CORRESPONSAL

¡AQUI LONDRES!

NUEVO

PON A PRUEBA TU HABILIDAD CON GYROSCOPE

MICROFILE

COMO PASAR A MICRODRIVE LOS PROGRAMAS "TURBO"

HOBBY PRESS S.A.

PONTE EN LA ONDA DE SILICON VALLEY

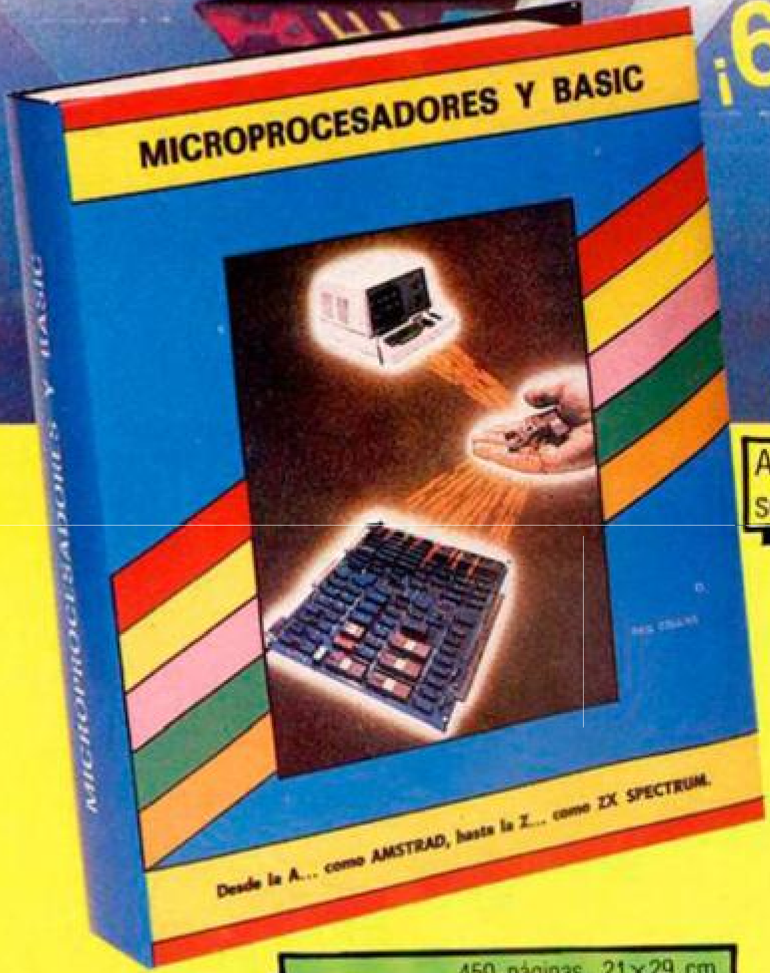
¿Aprender la Informática? ¡Sí! pero aprende lo de hoy y lo de mañana, ¡no lo de ayer! **«Microprocesadores y BASIC»** es el libro de hoy y de mañana. Da un salto hacia el futuro.

LAS MEJORES RAZONES DEL MUNDO PARA PEDIR HOY MISMO SU OBRA DE REFERENCIA

Una super obra por su importancia en el mercado de la edición especializada y dedicada a la Informática.

¡65 Ptas. el programa!

CONECTATE EN PARALELO RS232C CON LA UNIDAD DE MASA 720 Ko QUE CONTIENE **«MICROPROCESADORES Y BASIC»**





¡NUEVO EN ESPAÑA!

Ahora, puedes poner en tu mano todos los secretos de los microprocesadores y del BASIC

- 100 programas BASIC, originales e inéditos. Juegos, enseñanza, de utilidad, profesionales...
- Traducción y adaptación de tus programas. 100 instrucciones BASIC bajo la lupa.
- Guía de los microordenadores: 210 ordenadores individuales, familiares, semiprofesionales y profesionales con todas las características comparadas.
- Guía de las impresoras: 164 impresoras comprobadas y comparadas con todas sus características.
- Banco de pruebas: 45 ordenadores comprobados y comparados. Resultados.
- Cuadro comparativo de todos los BASIC que permite utilizar los programas con: Alice - Apple - Atari 400 - Atom - BBC - Commodore - DAI - Dragón 32 - Hector - Lynx - MZ80A - ORIC - PET/CBM - T07 - T199 - TRS80 - VIC 20 - ZX81 - ZX Spectrum...
- Interfaces para realizar por sí mismo, descripción completa de los montajes. La solución económica: 10 interfaces de fácil realización.
- Realización de los programas: desde la A... hasta la Z...
- ¿Cómo ganar dinero con su ordenador? 100 ideas para rentabilizar su microordenador.
- Todos los trucos empleados por los piratas de programas.
- Montar su microordenador en Kit: consejos y trucos útiles.
- Los genios de la programación. Cómo se lo montan para realizar, proteger y comercializar sus programas.
- Diccionario de la Informática.
- Cuáles serán las especialidades más interesantes dentro de los 5 próximos años.
- ¿Qué vale un microordenador? Hemos desmontado varios, hemos sumado parte por parte lo que compone el dentro de cada ordenador. Resultado: un microordenador se vende 5 ó 6 veces más caro que su precio de coste.
- Las mejores direcciones para comprar programas, periféricos, ordenadores, informarse de los «últimos secretos», etc...

450 páginas, 21×29 cm
Presentado en archivador plastificado, a todo color, con mecanismo de anillas y hojas móviles

Un enorme libro que explica todo, de forma sencilla, precisa, práctica, a cerca de los microprocesadores, ordenadores periféricos, accesorios y de la programación BASIC

UNA OBRA ECONOMICA: por su presentación, su contenido, los montajes de gran calidad técnica, con sólo realizar un interface universal 10 entradas-salidas ya tiene amortizado el precio del libro y ¡ha ganado dinero!

¡ADEMAS!

1/ventaja: posibilidad de comprar a precio sin competencia el material completo en Kit para construir los montajes.

2/ventaja: Una formidable documentación técnica, siempre al día con nuestro sistema exclusivo de puesta al día por medio de hojas móviles actualizando continuamente esta gran obra. (1 envío cada 2 meses de unas 100 páginas que se intercalan de un simple gesto en el archivador.)
Los envíos de puesta al día tratan y presentan las mejores novedades internacionales, nuevas técnicas, componentes, montajes, etc... siempre una suma de informaciones que le permitirá ahorrar un montón de dinero y de tiempo.



¿COMO GANAR UN MICROORDENADOR?

(TOSHIBA «PAP» - 8088 - 16 bits)

Simplemente rellenando el cupón numerado que encontrarás al final del libro **«Microprocesadores y BASIC»** y enviándolo a la editorial, entrarás en el sorteo que se celebrará ante notario el 1-6-86 en Barcelona. El ganador de este fabuloso regalo (**1 TOSHIBA «PAP»**, valorado en 350.000 Ptas.), lo recibirá en su domicilio por medio de transportista, en caja asegurada.
No dejes de participar, el super ordenador TOSHIBA «PAP» puede ser tuyo. **¡Suerte!**

¡REGALO DE FIN DE AÑO!

Para los «fans» de la microinformática:
Regalamos los fotolitos de todos los interfaces, periféricos, y accesorios descritos en la obra para realizar directamente los circuitos impresos de los montajes.

SOLICITUD DE PEDIDO

Editorial ALANPRESS, S.A. - Diputación 240, 6º, 7ª - 08007 Barcelona

☐ Sí, sírvanse enviarme... ejemplar de la obra en archivador con hojas móviles:

«MICROPROCESADORES Y BASIC»

al precio excepcional de lanzamiento de: 6.100 Ptas.
Pago por ☐ talón ☐ giro ☐ cheque

☐ Prefiero pagar contra reembolso de: 6.500 Ptas. (6.100 + 400 Ptas. de gastos)

☐ Deseo recibir una documentación

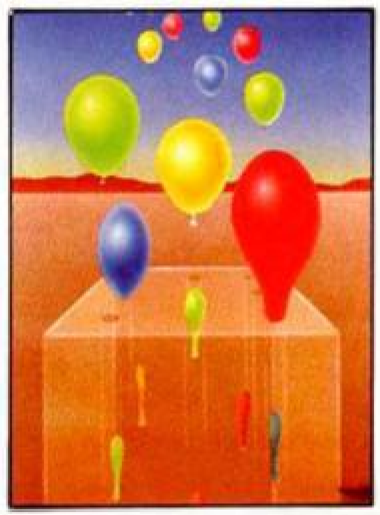
Nombre .... Apellidos ....
Calle .... Nº ....
Ciudad .... Cód. postal .... Provincia ....

Remita a **Editorial ALANPRESS, S.A. - Diputación, 240, 6º, 7ª 08007 Barcelona**

MICROHOBBY ESTA SEMANA

AÑO II. N.º 55. 3 al 9 de diciembre de 1985
125 ptas. (Sobretasa Canarias 10 ptas.)





Una rutina para descomprimir pantallas. Pág. 28.

PREMIADOS HOBBY-SUERTE

CARLOS MATEOS SEPEDA. Asunción, 37, 2.º Iz. (SEVILLA).
Cinta de programas (5.ª Cat.)

ESTEBAN BARRIOS LAS PERAS. Gral. Yagüe, 23, 11.º B. (BURGOS).
Cinta de programas (5.ª Cat.)

PABLO GONZALEZ FERNANDEZ. Concejo, 32, 7.º C. (ORENSE).
Cinta de programas (5.ª Cat.)

RICARDO MUÑOZ MUNICIO. Avda. Mediterráneo, 47. (MADRID).
Cinta de programas (5.ª Cat.)

JUAN DE DIOS AREVALO ROMAN. Avda. Andalucía, 64. (CADIZ).
Cinta de programas (5.ª Cat.)

ANTONIO ABAD GOULSA. Canónigo Montañés, 9, 2.º Manresa (BARCELONA).
Cinta de programas (5.ª Cat.)

OSCAR ESCUDERO TOLO. Salamanca, 25, Bj-A. Alcalá de Henares (MADRID).
Cinta de programas (5.ª Cat.)

JESUS SALAFRANCA ALVAREZ. Avda. Gral Villalba, 19, 3.º D. (OVIEDO).
Cinta de programas (5.ª Cat.)

JOSE LUIS VERA DIAZ. Bda. La Granja. Pza. Algodonales, 8. Jerez de la Frontera (CADIZ).
Un Joystick con su Interface (3.ª Cat.)

Director Editorial
José I. Gómez-Centurión

Director Ejecutivo
Domingo Gómez

Asesor Editorial
Gabriel Nieto

Redactor Jefe
Africa Pérez Tolosa

Diseño
Rosa María Capitel

Redacción
Amalio Gómez, Pedro Pérez, Jesús Alonso

Secretaría Redacción
Carmen Santamaría

Colaboradores
Primitivo de Francisco, Rafael Prades, Miguel Sepúlveda, Sergio Martínez y J. M. Lazo

Fotografía
Javier Martínez, Carlos Candel

Portada
José María Ponce

Dibujos
J. R. Ballesteros, A. Perera, F. L. Frontán, Pejo, J. M. López Moreno J. Igual, J. A. Calvo, Lóriga, J. Olivares

Edita
HOBBY PRESS, S. A.

Presidente
María Andrino

Consejero Delegado
José I. Gómez-Centurión

Jefe de Publicidad
Marisa Esteban

Publicidad Barcelona
José Galán Cortés
Tels.: 303 10 22 - 313 71 76

Secretaría de Dirección
Marisa Cogorro

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

Redacción, Administración y Publicidad
La Granja, s/n
Polígono Industrial de Alcobendas
Tel.: 654 32 11
Telex: 49480 HOPR

Dto. Circulación
Carlos Peropadre

Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
Rotedic, S. A. Ctra. de Irún, km. 12,450 (MADRID)

Fotocomposición
Espacio y Punto, S. A.
Paseo de la Castellana, 268

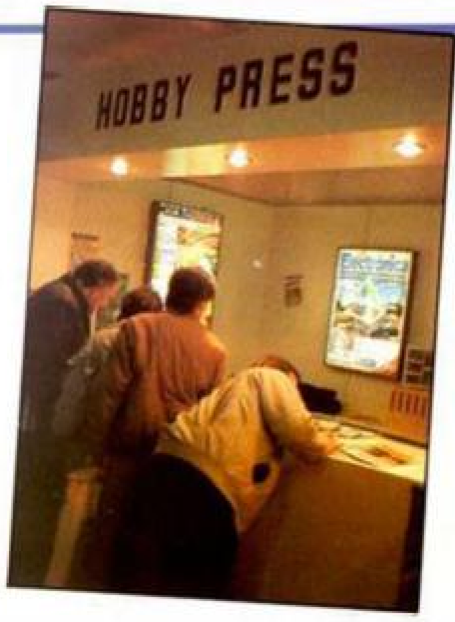
Fotomecánica
Grof
Ezequiel Solana, 16

Depósito Legal
M-36.598-1984

Representante para Argentina, Chile, Uruguay y Paraguay, Cía. Americana de Ediciones, S.R.L. Sud América 1.532. Tel.: 21 24 64. 1209 BUENOS AIRES (Argentina).

Solicitado control OJD

25 AÑOS DE S.I.M.O.



Como cada año por estas fechas, miles de personas se han reunido en la ya tradicional cita con la informática. El S.I.M.O., que cumple en esta ocasión 25 años de existencia, ha acogido en sus cada vez más nutridos pabellones, a todas aquellas personas que por una u otra razón están interesadas en el mundo de los ordenadores y la programación.

Por allí han desfilado desde expositores hasta importantes hombres de negocios, pasando por estudiantes, programadores, ejecutivos y hasta algún que otro curioso, todos ellos movidos por muy diversas razones, pero bajo el mismo denominador común: la informática.

Por supuesto nosotros no podíamos faltar a tan importante cita. Y allí estuvimos, en nuestro pequeño pero nutrido stand, donde tuvimos la oportunidad de contactar con algunos de vosotros, y donde pudimos recoger vuestras impresiones sobre la revista. Pero entre charla y charla, nos recorrimos la exposición con el fin de recoger aquellos temas que pensamos, pudieran ser de vuestro interés.

La verdad es que esta edición del S.I.M.O. no se ha caracterizado por la presentación de muchas novedades, pero de entre ellas podemos destacar dos de las que se puede decir que han sido prácticamente las estrellas en lo relativo a los microprocesadores. Estas son el 128 de Sinclair y el Atari 520 ST.



El punto más destacable a cerca del 128 Spectrum, que por cierto ya se encuentra a la venta en España, es que se han realizado unas unidades de disco específicas para este modelo, el cual funcionará con discos de 3,5 pulgadas. Estas unidades de discos han sido desarrolladas íntegramente en Portugal.

La otra gran novedad es la presentación por parte de Investrónica (que como sabéis es la distribuidora de Sinclair en nuestro país), del nuevo modelo de Atari, el 520 ST. Este ordenador que se adquiere con unidad de disco, monitor en blanco y negro y ratón, tiene grandes espectativas de ventas en España, pero presenta el inconveniente de que por el momento está arropado con una cantidad de software bastante reducida.

Y estos son los únicos temas dignos de mención, pues la gran mayoría de los stands no ofrecían ningún producto novedoso, y tan solo tenían la función de servir como escaparate para mostrar sus artículos y dar a conocer al público sus prestaciones específicas.

Un año más que el S.I.M.O. transcurre sin pena ni gloria.



UN MALAGUEÑO CONSIGUE EL PORSCHE OFRECIDO POR EL CONCURSO GYRON

Los ingleses no se lo acaban de creer. Les parecía imposible que un español, de Málaga para más señas, hubiera sido el campeón del concurso realizado entre las 162 personas de toda Europa que habían conseguido descubrir el código secreto del juego Gyron.



Y tanto fue así que tuvieron que venir personalmente a España para comprobar cómo J. Manuel Pérez Vázquez conseguía realizarlo en tan sólo 18 minutos, mientras que el segundo clasificado tardó en hacerlo 1 hora y 35 minutos.

El premio que va a obtener el hábil jugador por su increíble proeza será un Porsche 924 o bien su equivalente en dinero, lo que representa un verdadero record en cuanto al valor material de un regalo concedido en un concurso de ordenadores.

SINCLAIR ENIGMA

Todavía resuenan los ecos del 128 K cuando Sinclair Research vuelve a sorprendernos con el más osado de sus proyectos: el Sinclair ENIGMA.

Esta prodigiosa máquina (al menos sobre el papel) estará dotada de los últimos avances de la microinformática y, a groso modo, las características son:

— 1024 K de memoria RAM (un megabyte)

— Un par de unidades de disco de 3,5 pulgadas integradas (adiós a los microdrives).

— Los programas QUILL, EASEL, ARCHIVE y ABACUS, en su correspondiente versión, integrados en ROM.

— Ventanas, Iconos, Ratón incorporado, Alta Resolución en color y otras lindezas por el estilo estarán presentes en este Megacomputador.

Las primeras previsones apuntan hacia un lanzamiento al mercado para mayo del próximo año como equipo integrado: Monitor, teclado, unidades de disco, Ratón, software e impresora...! por un precio que oscilará entre 500 y 1.000 libras.

CITIZEN PRESENTA EN ESPAÑA SUS NUEVOS MODELOS DE IMPRESORAS

Citizen Europe Ltd., la filial europea de la prestigiosa compañía japonesa, ha efectuado la presentación para el mercado español de sus primeros modelos de impresoras. Estas son: MSP-10, MSP-15, MSP-20 y MSP-25, Premiere 35 y Citizen 120-D.

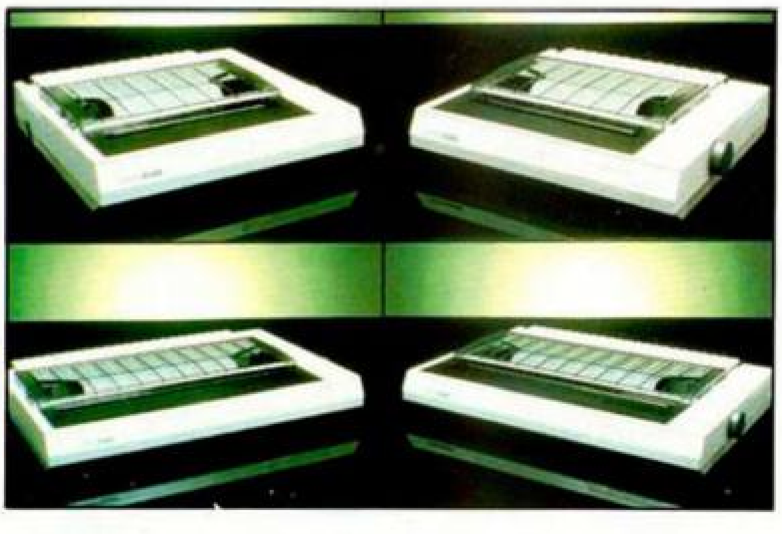


Sus velocidades de impresión oscilan entre los 160 y 200 caracteres por segundo, y unos anchos por columna entre 80 y 136 caracteres.

Otras características destacables son: la compatibilidad con Epson e IBM, sistemas de arrastre por empuje, sistema de autocomprobación y sensor de final de papel.

Los precios varían según el modelo, y éstos están comprendidos entre las 65.000 ptas. de la Citizen 120-D y las 195.000 ptas. de la MSP-25.

Todos estos modelos presentan, además, la ventaja de ofrecer dos años de garantía total a los usuarios.



AQUI LONDRES

La famosa compañía de software Firebird se dividirá en un período muy corto de tiempo en dos. Esta nueva filial llevará el nombre de Rainbird, y como primera misión tendrá que encargarse de finalizar y dar los últimos retoques a algunos títulos iniciados por Firebird.

El modelo PCW 8256 de Amstrad está teniendo un éxito arrollador en Gran Bretaña, hasta el punto de que en un gran número de tiendas de ordenadores tienen totalmente agotadas sus existencias de dicho modelo.

De entre las numerosas novedades de juegos, las que acaparan una mayor atención actualmente por parte del público inglés son:

— Gyroscope (Melbourne House)

— I, of the Mask (Electric Dreams), con cuyo programador, Sandy White, podremos ofreceros próximamente una interesante entrevista.

— Hi-Rise (Bouble Buf, una nueva casa de software que está teniendo un notable éxito en el mercado).

— Sir Fred (Mikro-Gen), programa con el que se está realizando una gran campaña publicitaria y que ha sido creado por una casa de software española.

Commodore ha puesto recientemente en el mercado el nuevo C128, el cual lleva incluido una unidad de disco. Su valor es de 800 libras. Este modelo es totalmente compatible con el C64.

Los títulos más vendidos actualmente en Inglaterra son:

Spectrum:

Monty on the run (Gremlin Grafics).
Impossible Mision (U.S. Gold).
Starquake (Bouble Buf)

Commodore:

Winter Games (U.S. Gold).
Monty on the run (Gremlin Grafics).
Fighting Warrior (Melbourne House).

Amstrad:

Finders Keepers (Master Tronicks).
Nonterraqueouus (Master Tronicks).
Soul of a robot (Master Tronicks).

Alan HEAP





¡¡¡ACCION!!!
COMMANDO
¡Por fin en España el programa más esperado de los últimos tiempos!
¡Atención al más sorprendente n.º 1 mundial!
SUPER COMBATE DE CHOQUE. EL COMANDO LUCHA EN UNA BATALLA UNICA CONTRA UN ENEMIGO EN SUPERIORIDAD. TODA LA ACCION Y TENSION DEL MEJOR DE LOS JUEGOS.
DISPONIBLE EN SPECTRUM COMMODORE 64 AMSTRAD
ZAFIRO SOFTWARE DIVISION
Paseo de la Castellana, 141. 28046 Madrid.
Tel. 459 30 04.
Telex: 22690 ZAFIR E
Editado,
reservados.
elite

EL ALMACEN

Juan José EPALZA



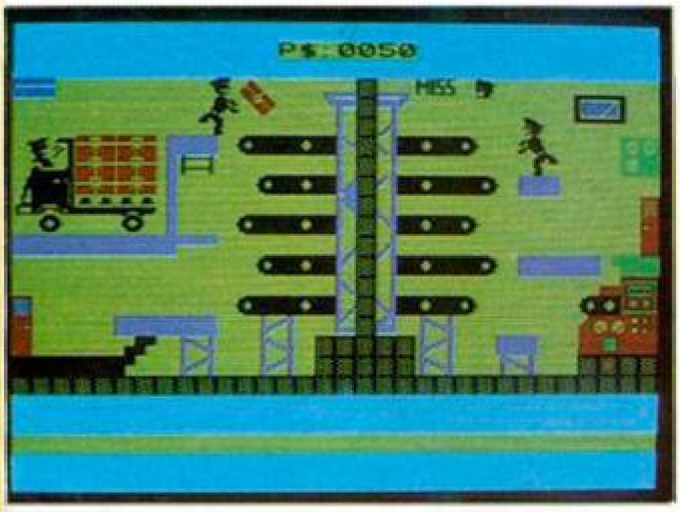
Javier Igual

TODAS LA LETRAS MAYUSCULAS SUBRAYADAS DEBERAN TECLEARSE EN MODO GRAFICO.

Spectrum 48 K

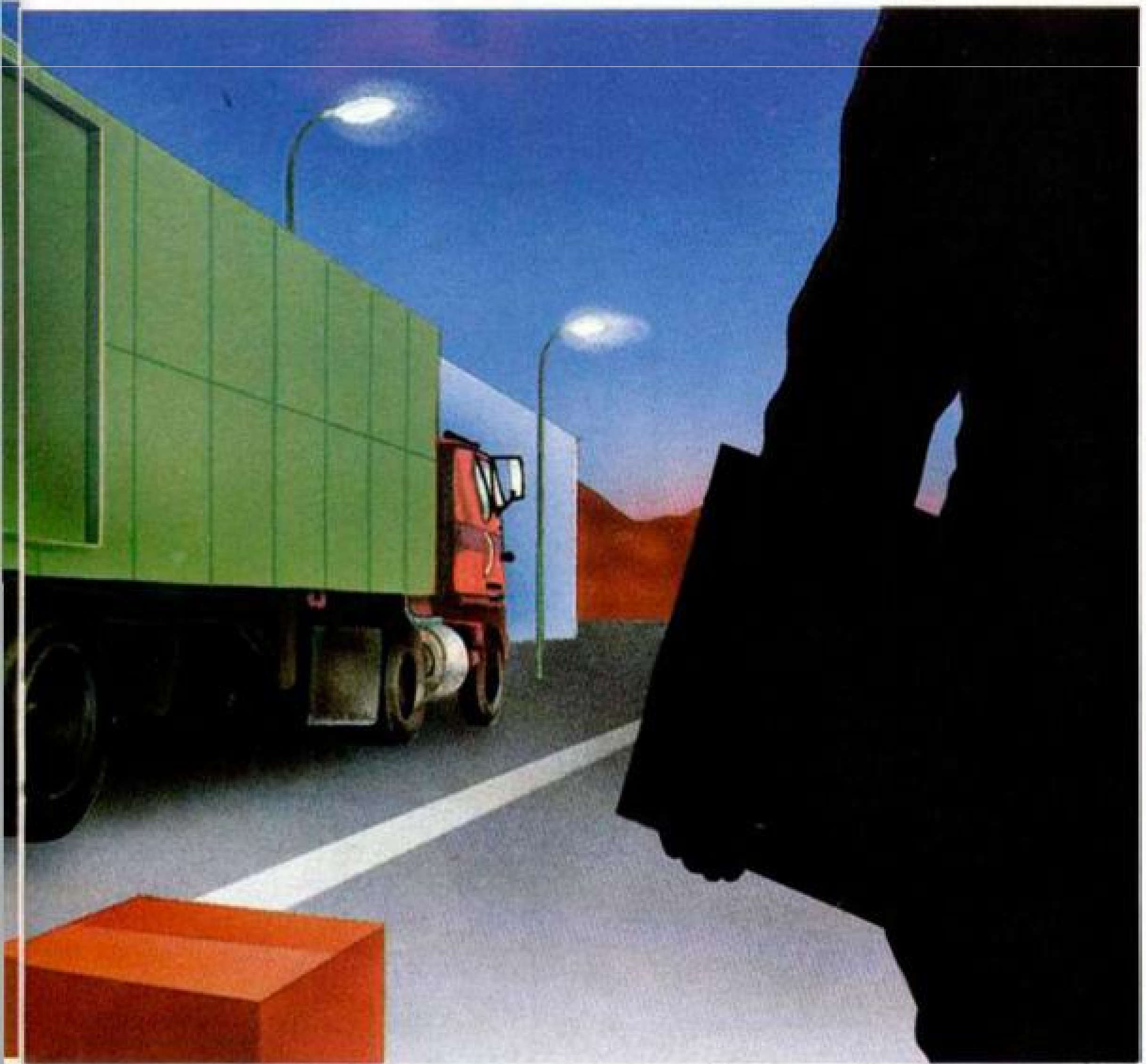
Es un día normal en la jornada laboral de nuestros Mario y Luilli, salvo en un pequeño detalle: el mal humor del jefe puede traer malas consecuencias.

Trabajan en una bodega y su misión es la de colocar y transportar cajas de vino en un camión. Pero hoy no pueden fallar una, ya que si dejan caer alguna caja serán despedidos automáticamente. Nosotros tenemos que ayudarles para que esto no ocurra teniendo en cuenta que por cada caja que recojamos nos darán un punto, dos por cada una que carguemos en el camión y diez por cada camión que logremos llenar. Si se nos caen tres cajas, no podremos seguir trabajando más.



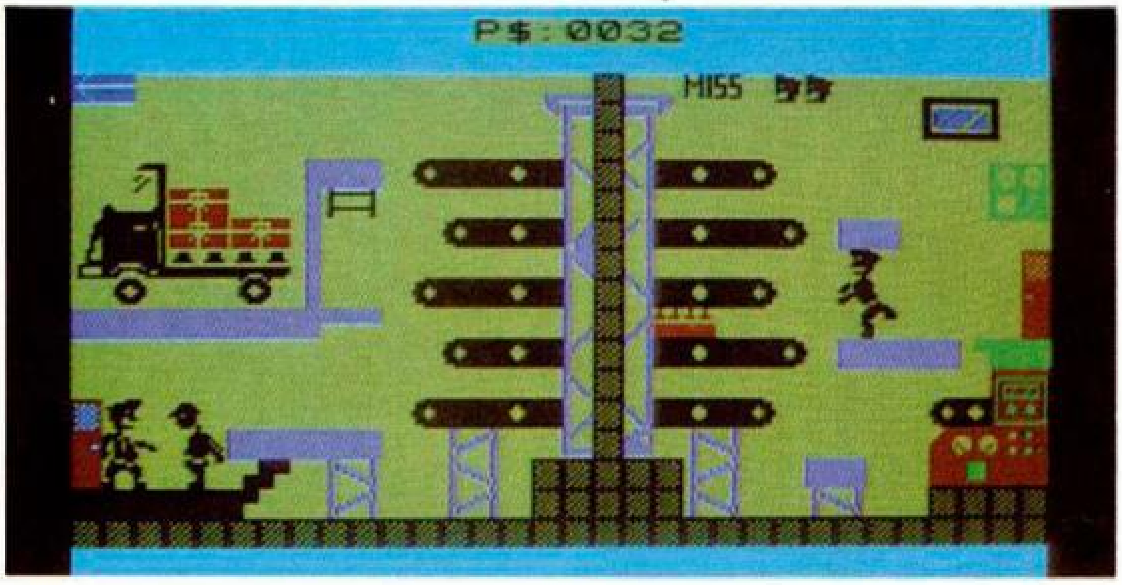

```
   1 CLEAR 63846  LET c=0  LET c
1=0  LET b=0  LET f=63847
   2 READ a  IF a=-2 THEN GO TO
5
   3 IF a=-1 THEN GO TO 7
   4 POKE f,a  LET f=f+1  LET c=
c+a  GO TO 2
   5 CLS  PRINT "PULSA una tecl
a para GRABAR"  SAVE "MARIO 1"CO
DE 63847,1688  STOP
   7 IF c<>c1 THEN PRINT FLASH 1
;"ERROR EN LINEA ";b  STOP
   8 LET c=0  READ c1  LET b=b+1
0  PRINT "VOY POR LA LINEA ";b
GO TO 2
   9 DATA -1,17188
  10 DATA 31,127,119,227,a,119,1
27,31,255,a,231,195,a,231,255,a,
248,254,238,199,a,238,254,248,32
,63,32,a,a,63,32,a,4,252,4,a,a,2
52,4,a,0,a,a,a,a,a,3,a,0,a,a,a,a
,a,255,a,0,a,a,a,a,a,192,a,3,a,0
,a,a,a,a,a,255,a,0,a,a,a,a,a,192
,a,0,a,a,a,a,a,192,a,a,a,a,a,a,a
,3,a,a,a,a,a,a,a,0,61,123,119,11
1,95,63,0,a,248,242,230,206,158,
62,0,255,171,213,171,213,171,213
,171,255,a,a,a,249,255,a,a,a,a,a
,a,159,255,a,a,0,a,a,a,a,a,31,a,
0,a,a,a,a,a,248,a,120,112,96,64,
0,a,a,a,a
  19 DATA -1,15569
  20 DATA 255,a,a,3,a,a,a,0,255,
a,a,192,a,a,160,30,14,6,2,0,a,a,
a,176,216,204,198,195,193,192,a,
3,a,a,131,195,99,51,27,15,5,3,a,
a,a,5,15,192,a,a,a,a,a,a,a,a,193
,195,199,207,223,255,159,27,51,9
9,195,131,3,a,a,a,a,a,a,a,a,131,
195,159,255,a,223,207,199,195,19
3,192,a,a,a,a,a,a,a,99,51,27,13,
7,3,a,a,a,a,a,7,13,27,51,99,192,
a,a,a,a,193,195,198,204,216,176,
224,192,a,a,a,195,131
  29 DATA -1,16054
  30 DATA 3,a,a,a,a,a,a,192,a,160,
240,216,204,198,195,3,131,195,99
,51,27,15,5,3,a,a,a,a,a,7,15,192
,a,a,a,193,195,198,204,216,240,1
60,192,a,255,a,0,31,63,121,249,2
55,a,a,0,195,131,3,a,a,a,a,a,48,
44,47,51,48,a,a,a,6,a,a,198,246,
58,10,6,48,a,a,49,55,46,40,48,6,
26,122,230,134,6,a,a,255,a,128,1
91,160,173,171,160,255,a,1,253,5
,213,181,5,191,187,177,a,187,191
,128,255,253,221,141,a,221,253,1
,255,15,31,62,124,253,252,254,25
5,a,a
  39 DATA -1,20303
  40 DATA 60,24,a,153,126,255,a,
249,121,63,191,57,121,255,a,159,
155,255,a,155,159,255,a,227,193
201,a,193,227,255,a,199,131,147,
a,131,199,255,a,193,a,a,a,a,255,
a,a,231,255,a,191,243,255,a,0,6,
.127,3,67,147,35,67,0,a,a,a,64,6
24,255,a,131,3,a,a,a,7,255,a,1,
,a,a,7,6,4,a,255,251,27,253,a,a
a,a,12,8,a,15,63,32,a,63,255,a,a
,a,240,239,216,240,253,a,a,a,63,
223,115,31,0,28,a,62,0,255,a,a,0
,28,a,62,0,255,a,248,0,28,a,62,0
,255,a
  49 DATA -1,14270
  50 DATA 126,7,15,28,24,28,15,7
,0,128,192,224,96,224,192,128,0,
23,15,28,24,28,15,7,0,160,192,22
4,96,224,192,128,0,68,a,238,0,25
5,a,a,a,85,a,238,0,255,a,a,a,a,a
,a,0,255,a,a,a,127,124,125,0,126
,125,127,a,a,31,95,0,63,95,127,a
,4,14,31,47,119,251,125,62,0,a,a
,128,192,224,240,248,31,15,7,3,1
,0,a,a,124,190,223,238,244,248,1
12,32,2,12,4,32,113,248,124,62,0
,a,a,64,128,a,8,48,16,129,198,22
6,240,248,112,32
  59 DATA -1,9455
  60 DATA 0,a,a,64,128,a,0,a,a,1
28,192,224,240,248,112,32,0,a,a,
32,112,248,124,62,2,12,8,32,113,
248,124,62,0,128,0,64,144,160,8,
50,20,129,198,226,240,248,112,32
,a,112,248,244,238,223,190,124,0
,a,a,1,3,7,15,31,248,240,224,192
,128,0,a,a,62,125,251,119,47,31,
14,4,64,48,32,4,142,31,62,124,0,
a,a,2,1,a,16,12,8,129,99,71,15,3
1,14,4,0,a,a,2,1,a,0,a,a,1,3,7,1
5,31,14,4,0,a,a,4,14,31,62,124,6
4,48,160,4,142,31,62,124,0,1,0,2
,9
  69 DATA -1,7869
  70 DATA 5,16,76,40,129,99,71,1
5,31,14,4,0,4,14,15,55,123,129,6
2,0,a,a,a,128,192,a,48,28,14,5,3
,1,0,a,a,164,60,94,238,244,248,1
12,32,29,60,122,119,47,31,14,4,5
6,112,160,192,128,0,a,a,a,a,a,a,
1,3,a,12,0,32,112,240,236,222,19
0,124,138,a,216,170,135,a,a,0,23
8,136,a,238,34,a,238,0,2,7,15,31
,15,71,11,17,0,a,146,196,232,0,2
4
  79 DATA -1,5455
  80 DATA 166,72,188,18,136,43,1
6,37,200,32,135,63,a,31,95,127,1
12,0,a,a,1,0,48,120,252,0,8,0,a,
a,66,7,15,254,127,62,92,9,0,36,0
,31,63,126,28,72,4,32,0,a,124,0,
a,14,0,a,a,a,15,18,216,0,124,7,0
,a,4,250,18,164,196,56,0,a,7,8,4
,8,4,9,16,240,24,136,68,a,36,168
,136,16,4,26,37,209,38
  89 DATA -1,9383
  90 DATA 152,96,68,132,a,232,40
,16,96,128,0,28,120,232,252,248,
224,120,0,a,1,3,0,a,a,a,48,124,2
22,15,79,224,255,a,3,31,127,61,6
3,57,24,31,192,a,128,0,a,128,0,a
,8,120,246,218,155,a,223,159,0,a
,a,224,120,12,128,a,35,60,127,11
3,96,32,224,a,0,128,192,a,a,128,
224,a,0,a,12,120,224,0,128,a,0,a
,48,30,7,0,1,a,0,1,3,a,a,1,7
  99 DATA -1,5725
 100 DATA 7,196,60,254,142,6,4,7
,a,0,a,a,7,30,48,1,a,16,30,111,9
1,217,a,251,249,3,a,1,0,a,1,0,a,
192,248,254,188,252,156,24,248,0
,56,76,164,132,104,36,16,28,36,8
8,37,17,36,8,16,7,3,a,2,7,3,1,3,
0,192,224,240,224,a,a
 109 DATA -1,9678
 110 DATA 192,0,1,3,a,a,7,a,1,12
3,224,240,232,244,250,236,198,6,
7,3,1,0,a,3,a,36,224,192,a,a,a,a
,a,0,14,31,63,119,31,a,14,0,a,a
128,0,a,a,128,0,a,a,1,0,a,a,1,0,
112,248,252,238,248,a,112,100,7,
3,a,a,a,a,a,96,224,192,128,0,a,1
92,a,1,7,15,23,47,95,55,99,0,128
,192,a,a,224,a,128,0,3,7,15,7,a,
a,3,224,192,a,64,224,192,128,192
,0,a,2,7,15,29,63,30,0,12,18,49
 119 DATA -1,8356
 120 DATA 37,130,52,136,15,3,1,a
,a,3,6,12,128,0,192,248,250,247,
47,31,0,a,a,a,a,12,252,140,0,a,a
,a,a,48,63,49,1,0,3,31,95,239,24
4,248,240,192,128,a,a,192,96,48,
0,48,72,140,164,65,44,17,0,a,64,
224,240,184,252,120,0,a,a,31,68,
115,255,254,0,a,a,a,a,1,a,6,120,
62,55,126,255,253,248,16,0,a,224
,48,24,128,192,96,7,a,15
 129 DATA -1,10643
 130 DATA 15,14,4,28,a,224,a,240
,a,112,32,56,a,0,a,7,12,24,1,3,6
,30,124,28,126,255,191,31,8,0,a,
a,a,a,128,a,96,0,a,a,248,124,110
,255,127,0,1,a,a,3,a,4,31,224,24
4,246,243,251,251,195,0,63,115,1
13,0,1,a,a,a,128,192,a,a,a,128,1
92,a,1,3,a,a,a,1,3,a,252,206,142
,0,128,a,a,a,7,47,111,207,63,231
,195,0,a,128,a,a,192,a,32,248,0,
a,a,3,1,a,0,a,28,120,248,232,252
 139 DATA -1,9257
 140 DATA 224,248,64,56,30,31,23
,63,7,31,2,0,a,a,192,128,a,0,a,a
3,7,3,1,a,0,a,248,a,208,240,248
,128,240,64,31,a,11,15,31,1,15,2
,0,192,224,192,128,a,0,a,3,1,a,3
,1,0,1,0,240,120,220,248,a,112,2
24,64,0,1,0,a,1,29,249,192,112,2
48,252,222,238,60,156,216,0,56,3
1,0,a,a,a,a,120,32,248,252,a,251
,103,31,0,a,a,a,48,a,240,0,28,a,
24,3,a,a,56,a,0
 149 DATA -2
```

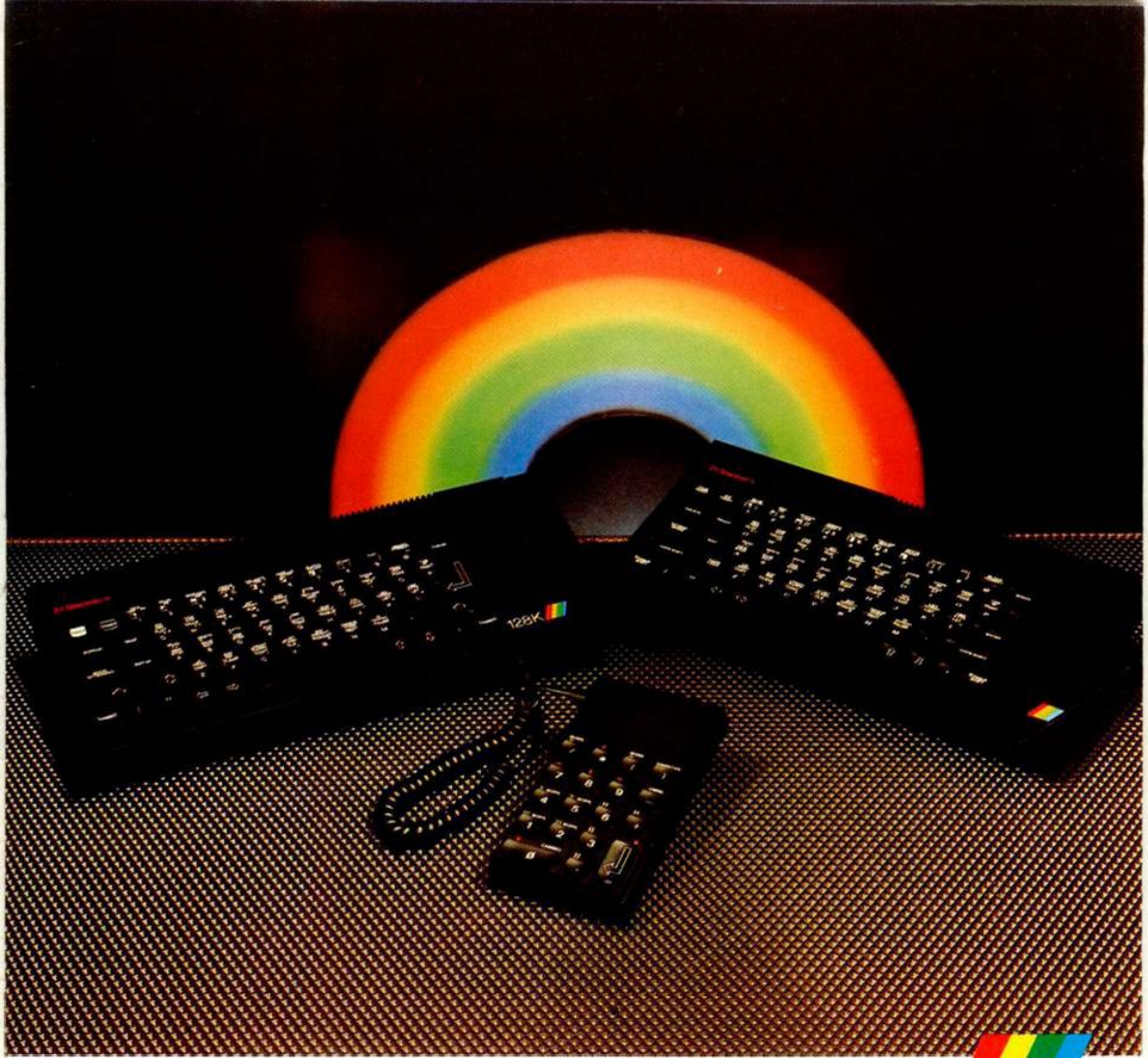
```
   1 LET rec2=0  LET rec=0
   2 DIM c(4)  LET cam=0  LET ma
x1=32  LET col=1  LET max=32  LE
T p1=0  LET p2=0  LET c2=2  LET
ca=0  LET cs=0  LET p=0  LET c1=
0  LET c=1  LET m=2  LET l=3  LE
T mi=0  LET ct=0
   3 LET c(3)=-18  LET c(4)=-25
```



```
LET c(1)=1  LET c(2)=-3
5 BORDER 5  PAPER 6  INK 0  CLS  FOR f=19 TO 21  PRINT AT f,0; INK 5;"[redacted]";AT f-19,0;"[redacted]"  NEXT f
6 PRINT AT 1,13;"P$ 0000";#0;"RECORD= P$  ";"00000"( TO 4-LEN STR$ (c(c));c(c;"  CAMIONES  ";c<2  FOR f=23264 TO 23295  POKE f,22  NEXT f
10 INK 3  PRINT AT 15,5;"[redacted]"  FOR f=0 TO 7  PRINT AT 11,f;"[redacted]"  NEXT f  FOR f=10 TO 6 STEP -1  PRINT AT f,7;"[redacted]"  NEXT f  PRINT AT 6,8;"[redacted]"  PRINT AT 11,8;"[redacted]"  INK 0  PRINT AT 17,0;"[redacted]";AT 17,5;"[redacted]";AT 16,5;"[redacted]"
11 PLOT 15,148  DRAW -14,0  DRAW 3,3  PLOT 1,148  DRAW 4,-4  POKE 23606,95  POKE 23607,248  PRINT AT 16,8; INK 3;"NO";AT 17,8;"PO"  PRINT AT 14,0; INK 2; PAPER 5;"0""1""[redacted]"  PRINT AT 7,8; INK 0;"$%"
12 PRINT AT 3,0; OVER 1; PAPER 1; INK 7;"  "  PRINT AT 6,11;"[redacted]";AT 8,12;"[redacted]";AT 10,11;"[redacted]";AT 12,12;"[redacted]";AT 14,11;"[redacted]"
13 LET a=16  PRINT INK 3;AT 3,a-1;"3  4";AT 4,a-1;"87 65";AT 5,a;"  C";AT 6,a;"9  ";AT 7,a;" -";AT 8,a;"= >";AT 9,a;"@ ?";AT 10,a;"A B";AT 11,a;"D C";AT 12,a;"E F";AT 13,a;"G -";AT 14,a;"J H";AT 15,a;"K L"
14 PRINT INK 0;AT 16,15;"00000";AT 17,15;"00000";AT 18,0;"00000000000000000000000000000000"  FOR f=3 TO 18  PRINT AT f,17; INK 0;"0"  NEXT f
15 PRINT AT 6,19;"[redacted]";AT 8,19;"[redacted]";AT 10,19;"[redacted]";AT 12,19;"[redacted]";AT 14,19;"[redacted]"
16 PRINT AT 16,12; INK 3;"NO";AT 17,12;"PO";AT 15,12;"PO";AT 15,20;"NO";AT 16,20;"PO";AT 17,20;"NO"
17 PRINT INK 0;AT 3,27;"& (";AT 4,27;"-"; INK 1;"./"; INK 0;"";AT 5,27;"!***"
18 PRINT INK 3;AT 8,25;"[redacted]";AT 12,25;"[redacted]"; INK 4;"[redacted]"; INK 2;AT 9,31;"0";AT 10,31;"2";AT 11,31;"[redacted]"
21 PRINT AT 13,30; INK 2; PAPER 4;"RS";AT 14,30;"TU";AT 15,28; PAPER 6;"VUX"; PAPER 4;"Y";AT 16,28;"[redacted]"
22 PRINT AT 17,28;"0000"
23 PRINT AT 17,24; INK 3;"NO";AT 16,24;"[redacted]"  PRINT AT 14,28;"!""""
30 INK 6  PRINT AT 5,3;"pqpq";AT 6,2;"!pqpq";AT 7,1;"_£pqpq";AT 8,0;"a[redacted]bpqpq";AT 9,0;"cdefgh";AT 10,1;"ij";AT 10,5;"kl"  PRINT AT 6,30; INK 4;"Z[";AT 7,30;"\]"
40 POKE 23606,119  POKE 23607,251  PRINT AT 13,1;"ST";AT 8,29;"ST";AT 14,1;"EF(\";AT 15,1;"GHW x";AT 16,1;"IUYZ";AT 9,27;"cdOR";AT 10,27;"abOP";AT 11,27;"_£MN"
41 PRINT AT 6,25;"6"  POKE 23607,254  PRINT "  "  POKE 23607,251  PRINT AT 7,25;">";AT 13,8;"yz";AT 14,8;"{|";AT 4,9;"yz";AT 5,9;"{|";AT 9,8;"yz";AT 10,8;"{|"  PRINT AT 13,5;"e/";AT 14,5;"ghi";AT 10,25;"6"  POKE 23607,254  PRINT "  "  POKE 23607,251  PRINT AT 11,25;">"
42 PRINT AT 10,0;",";AT 10,3;"="  PRINT INK 6;AT 3,20;"12"; INK 6;"[redacted]RRR"
43 PRINT AT 13,25;"o";AT 14,24;"pq/";AT 15,24;"[!"
44 POKE 23607,254  PRINT AT 3,6;"+";AT 4,6;"-,";AT 5,7;"0"  PRINT AT 5,25;"#$";AT 8,8;"!+";AT 3,9;"%&";AT 12,8;"!""";AT 9,25;"!"
45 POKE 23606,95  POKE 23607,248  PRINT AT 11,19;"mmmm";AT 9,19;"mmmm";AT 9,12;"nnnn";AT 7,12;"nnnn";AT 7,19;"oooo";AT 5,19;"oooo";AT 5,12;"pqpq";AT 11,12;"[redacted]";AT 13,12;"[redacted]";AT 13,27;"[redacted]";AT 13,19;"[redacted]"
46 PRINT AT 10,23;"vw";AT 11,23;"!x";AT 6,23;"ri";AT 7,23;"tu"
47 POKE 23606,119  POKE 23607,251  PRINT AT 12,10;"#";AT 13,10;"$.";AT 8,10;"!";AT 9,10;"!."
48 PRINT AT 3,3;"/0";AT 4,3;"-.";AT 3,11;"!+";AT 4,11;"+";AT 16,10;"34";AT 17,10;"56";AT 16,22;"34";AT 17,22;"56";AT 16,26;"78";AT 17,26;"9"
50 POKE 23606,0  POKE 23607,60
60 PRINT INK 7; PAPER 1;AT 21,12;"CAMIONES  ";c:m  PRINT AT 19,2; PAPER 0; INK 8; INVERSE 1;"PULSA UNA TECLA PARA EMPEZAR"
65 IF INKEY$="" THEN RANDOMIZE USR 63781  GO TO 65
66 BORDER 5  PRINT AT 19,2; INK 8;"[redacted]"
67 INK 0  PRINT OVER 1;AT 4,0;"  ";AT 5,0;"  ";AT 6,0;"  ";AT 7,0;"  ";AT 8,0;"  ";AT 9,0;"  ";AT 10,1;"  ";AT 10,5;"  "
68 IF l=1 THEN PRINT AT 3,9; OVER 1; INK 0;"  ";AT 4,9;"  ";AT 5,9;"  "
69 IF l=2 THEN PRINT AT 8,8; OVER 1; INK 0;"  ";AT 9,8;"  ";AT 10,8;"  "
70 IF l=3 THEN PRINT AT 12,8; OVER 1; INK 0;"  ";AT 13,8;"  ";AT 14,8;"  "
71 IF m=1 THEN PRINT AT 5,25; OVER 1; INK 0;"  ";AT 6,25;"  ";AT 7,25;"  "
72 IF m=2 THEN PRINT AT 9,25; OVER 1; INK 0;"  ";AT 10,25;"  ";AT 11,25;"  "
73 IF m=3 THEN PRINT AT 13,24; OVER 1; INK 0;"  ";AT 14,24;"  ";AT 15,24;"  "
100 IF INKEY$="" THEN GO TO 140
101 IF INKEY$="q" THEN BEEP .02,40  LET l=l-(1 AND l>1)
110 IF INKEY$="a" THEN BEEP .02,40  LET l=l+(1 AND l<3)
111 IF INKEY$="l" THEN BEEP .02,40  LET m=m+(1 AND m<3)  GO TO 130
112 IF INKEY$="p" THEN BEEP .02,40  LET m=m-(1 AND m>1)  GO TO 130
122 PRINT AT 3,9; OVER 1; INK 6;"  ";AT 4,9;"  ";AT 5,9;"  ";AT 8,8;"  ";AT 9,8;"  ";AT 10,8;"  ";AT 12,8;"  ";AT 13,8;"  ";AT 14,8;"  "
123 IF l=1 THEN PRINT AT 3,9; OVER 1; INK 0;"  ";AT 4,9;"  ";AT 5,9;"  "
124 IF l=2 THEN PRINT AT 8,8; OVER 1; INK 0;"  ";AT 9,8;"  ";AT 10,8;"  "
125 IF l=3 THEN PRINT AT 12,8; OVER 1; INK 0;"  ";AT 13,8;"  ";AT 14,8;"  "
126 GO TO 140
130 PRINT AT 5,25; OVER 1; INK 6;"  ";AT 6,25;"  ";AT 7,25;"  ";AT 9,25;"  ";AT 10,25;"  ";AT 11,25;"  ";AT 13,24;"  ";AT 14,24;"  ";AT 15,24;"  "
131 IF m=1 THEN PRINT AT 5,25; OVER 1; INK 0;"  ";AT 6,25;"  ";AT 7,25;"  "
132 IF m=2 THEN PRINT AT 9,25; OVER 1; INK 0;"  ";AT 10,25;"  ";AT 11,25;"  "
133 IF m=3 THEN PRINT AT 13,24; OVER 1; INK 0;"  ";AT 14,24;"  ";AT 15,24;"  "
140 IF c1=c2 THEN LET c1=0
143 LET c1=c1+1
144 LET c(c1)=c(c1)+1  LET c=c(c1)  IF c>=33 THEN LET c(c1)=0
145 IF c=1 THEN PRINT AT 13,29; INK 2; OVER 1;"  "
146 IF c=2 THEN PRINT AT 13,28; INK 2; OVER 1;"  "
147 IF c=3 THEN PRINT AT 13,27; INK 2; OVER 1;"  "; INK 6;AT 13,29;"  "  GO SUB 2000  IF m<>3 THEN GO TO 5215
148 IF c=3 AND m=3 THEN GO TO 3000
149 IF c=4 THEN BEEP .004,60  PRINT AT 13,21; OVER 1; INK 2;"  "; INK 6;AT 13,27;"  "
150 IF c=5 THEN BEEP .004,60  PRINT AT 13,19; OVER 1; INK 2;"  "; INK 6;AT 13,21;"  "
160 IF c=6 THEN BEEP .004,60  PRINT AT 13,14; OVER 1; INK 2;"  "; INK 6;AT 13,19;"  "
170 IF c=7 THEN BEEP .004,60  PRINT AT 13,12; OVER 1; INK 2;"  "; INK 6;AT 13,14;"  "
180 IF c=8 THEN BEEP .004,60  PRINT AT 12,10; OVER 1; INK 2;"  ";AT 13,10;"  ";AT 13,12; INK 6;"  "  GO SUB 2000  IF l<>3 THEN LET c=12  GO TO 5100
190 IF c=9 THEN BEEP .004,60  PRINT AT 11,12; OVER 1; INK 2;"  "; INK 6;AT 12,10;"  ";AT 13,10;"  "
200 IF c=10 THEN BEEP .004,60  PRINT AT 11,14; OVER 1; INK 2;"  "; INK 6;AT 11,12;"  "
210 IF c=11 THEN BEEP .004,60  PRINT AT 11,19; OVER 1; INK 2;"  ";AT 11,14; INK 6;"  "
220 IF c=12 THEN BEEP .004,60  PRINT AT 11,21; OVER 1; INK 2;"  "; INK 6;AT 11,19;"  "
230 IF c=13 THEN BEEP .004,60  PRINT AT 10,23; OVER 1; INK 2;"  ";AT 11,23;"  ";AT 11,21; INK 6;"  "  GO SUB 2000  IF m<>2 THEN LET c=10  GO TO 5200
240 IF c=14 THEN BEEP .004,60  PRINT AT 9,21; OVER 1; INK 2;"  "; INK 6;AT 10,23;"  ";AT 11,23;"  "
250 IF c=15 THEN BEEP .004,60  PRINT AT 9,19; OVER 1; INK 2;"  "; INK 6;AT 9,21;"  "
260 IF c=16 THEN BEEP .004,60  PRINT AT 9,14; OVER 1; INK 2;"  "; INK 6;AT 9,19;"  "
270 IF c=17 THEN BEEP .004,60  PRINT AT 9,12; OVER 1; INK 2;"  "; INK 6;AT 9,14;"  "
280 IF c=18 THEN BEEP .004,60  PRINT AT 8,10; OVER 1; INK 2;"  ";AT 9,10;"  ";AT 9,12; INK 6;"  "  GO SUB 2000  IF l<>2 THEN LET c=8  GO TO 5100
290 IF c=19 THEN BEEP .004,60  PRINT AT 7,12; OVER 1; INK 2;"  "; INK 6;AT 8,10;"  ";AT 9,10;"  "
300 IF c=20 THEN BEEP .004,60  PRINT AT 7,14; OVER 1; INK 2;"  "; INK 6;AT 7,12;"  "
310 IF c=21 THEN BEEP .004,60  PRINT AT 7,19; OVER 1; INK 2;"  "; INK 6;AT 7,14;"  "
320 IF c=22 THEN BEEP .004,60  PRINT AT 7,21; OVER 1; INK 2;"  "; INK 6;AT 7,19;"  "
330 IF c=23 THEN BEEP .004,60  PRINT AT 6,23; OVER 1; INK 2;"  ";AT 7,23;"  ";AT 7,21; INK 6;"  "  GO SUB 2000  IF m<>1 THEN LET c=6  GO TO 5200
340 IF c=24 THEN BEEP .004,60  PRINT AT 5,21; OVER 1; INK 2;"  "; INK 6;AT 6,23;"  ";AT 7,23;"  "
```

```
 350 IF c=25 THEN BEEP .004,60
PRINT AT 5,19; OVER 1; INK 2;"
";AT 5,21; INK 6;"
 360 IF c=26 THEN BEEP .004,60
PRINT AT 5,14; OVER 1; INK 2;"
"; INK 6;AT 5,19;"
 370 IF c=27 THEN BEEP .004,60
PRINT AT 5,12; OVER 1; INK 2;"
"; INK 6;AT 5,14;"
 380 IF c=28 THEN BEEP .004,60
PRINT AT 3,11; OVER 1; INK 2;"
";AT 4,11;"    ";AT 5,12; INK 6;"
": GO SUB 2000: IF l=1 THEN LE
T c=21: GO TO 5100
 381 IF c<28 THEN GO TO 100
 382 IF col=1 THEN GO TO 500
 383 IF c=ma1 THEN LET ma=ma-
.5: LET ma1=ma1-1: LET col=1:
LET ca=ca+1
 384 IF ca=8 THEN GO TO 4900
 385 IF c=25 AND l=1 THEN LET ma
1=ma1-.5: PRINT OVER 1; INK 6;AT
 3,11;"  ";AT 4,11;"  ";AT 3,9;"
";AT 4,9;"  ";AT 5,9;"  "; INK
 0;AT 3,6;"  ";AT 4,6;"  ";AT
5,7;"  ": BEEP .2,60: PRINT AT 3
,6; OVER 1; INK 6;AT 3,6;"  ";A
T 4,6;"  ";AT 5,7;"  "; INK 0
;AT 3,9;"  ";AT 4,9;"  ";AT 5,9
 390 IF c=29 THEN BEEP .004,60
PRINT AT 5,5; OVER 1; INK 2;"
 400 IF c=30 THEN BEEP .004,60
PRINT AT 6,5; OVER 1; INK 2;"
";AT 5,5; INK 6;"
 410 IF c=31 THEN BEEP .004,60
PRINT AT 7,5; OVER 1; INK 2;"
";AT 6,5; INK 6;"
 420 IF c=32 THEN BEEP .004,60
PRINT AT 8,5; OVER 1; INK 2;"
";AT 7,5; INK 6;"
 425 IF ca=8 THEN GO TO 4950
 430 GO TO 100
 500 IF c<25 THEN GO TO 100
 503 IF c=ma1 THEN LET ma=ma+
5: LET col=2: LET ca=ca+1
 505 IF ca=8 THEN GO TO 4900
 510 IF c=28 AND l=1 THEN GO SUB
 2000: LET ma=ma-.5: PRINT OVE
R 1; INK 6;AT 3,11;"  ";AT 4,11;
"  ";AT 3,9;"  ";AT 4,9;"  ";AT
5,9;"  "; INK 0;AT 3,6;"  ";AT
4,6;"  ";AT 5,7;"  ": BEEP .2,6
0: PRINT OVER 1; INK 6;AT 3,6;"
  ";AT 4,6;"  ";AT 5,7;"  "; I
NK 0;AT 3,9;"  ";AT 4,9;"  ";AT
5,9;"  "
 520 IF c=29 THEN BEEP .004,60
PRINT AT 5,3; OVER 1; INK 2;"  "
 530 IF c=30 THEN BEEP .004,60
PRINT AT 6,3; OVER 1; INK 2;"  "
;AT 5,3; INK 6;"  "
 540 IF c=31 THEN BEEP .004,60
PRINT AT 7,3; OVER 1; INK 2;"  "
;AT 6,3; INK 6;"  "
 550 IF c=32 THEN BEEP .004,60
PRINT AT 8,3; OVER 1; INK 2;"  "
;AT 7,3; INK 6;"  "
 560 IF ca=8 THEN GO TO 4950
1999 GO TO 100
2000 LET p=p+1+(1 AND c$=1): IF
p>=9999 THEN LET p=0
2001 PRINT AT 1,16;"00000"( TO 4
-LEN STR$ p);p
2010 IF (p=500 OR (l=1 AND p=501
) OR p=3000 OR (l=1 AND p=3001))
 THEN LET p=p+2: BEEP .2,35: BEE
P .2,35: BEEP .2,35: LET mi=mi-2
: GO SUB 5300: LET c$=1
2999 RETURN
3010 POKE 23606,119: POKE 23607,
251: PRINT AT 13,27; INK 6;"▬";
 INK 0;AT 13,24;"x";AT 14,23; IN
K 0;"uvw"
3011 BEEP .004,60: POKE 23606,0:
 POKE 23607,60: PRINT AT 13,25;"
 ";AT 14,26;"  "; OVER 1; INK 2;
AT 13,21;"  ": BEEP .2,60
3020 POKE 23606,119: POKE 23607,
251: PRINT AT 13,21; INK 2;"▬";
AT 13,25; INK 0;"o";AT 14,24; IN
K 0;"pqr"
3030 POKE 23606,0: POKE 23607,60
: PRINT AT 13,24;"  ";AT 14,23;"
 "
3050 GO TO 150
3100 IF c2=1 THEN GO SUB 3150: R
ETURN
3120 LET c(c1)=-6
3140 RETURN
3150 IF cam=0 OR cam=1 OR cam=2
THEN LET c2=2
3160 IF cam=3 OR cam=4 OR cam=5
THEN LET c2=3
3180 IF cam>=6 THEN LET c2=4
3190 LET c(1)=1: LET c(2)=-3: LE
T c(3)=-18: LET c(4)=-25
3200 OVER 1: INK 6: PRINT AT 3,1
1;"  ";AT 4,11;"  ";AT 5,12;"
 ";AT 5,19;"  ";AT 6,23;"  ";A
T 7,23;"  ";AT 7,12;"  ";AT 7,
19;"  ";AT 8,10;"  "
3210 PRINT AT 9,10;"  ";AT 9,12;
"  ";AT 9,19;"  ";AT 13,10;"
 ";AT 12,10;"  ";AT 13,19;"
 ";AT 11,12;"  ";AT 11,19;"
 ";AT 10,23;"  ";AT 11,23;"
"
3500 OVER 0: INK 0: RETURN
4899 GO TO 100
4900 POKE 23606,119: POKE 23607,
251: PRINT AT 6,0;"CB";AT 7,1;"D
"
4905 POKE 23606,0: POKE 23607,60
4940 GO TO 385
4950 PRINT AT 6,0;"  ": POKE 236
07,248: POKE 23606,95: PRINT AT
7,1;"  "
4955 POKE 23606,0: POKE 23607,60
5000 LET cam=cam+1: PRINT INK 7;
 PAPER 1;AT 21,12;"CAMIONES ";c
am: LET ma=32: LET ma1=32: LET
 col=1: GO SUB 3150: LET ca=0: I
F p+10>=9999 THEN LET p=9999: GO
 TO 5002
5001 FOR f=p TO p+9+(10 AND c$=1
): LET p=p+1: PRINT AT 1,16;"000
00"( TO 4-LEN STR$ p);p: BEEP .0
1,40: NEXT f
5002 POKE 23606,119: POKE 23607,
251: FOR f=1 TO 100: PRINT OVER
1;AT 10,6;"f": PRINT OVER 1;AT 1
0,6;"T": NEXT f: POKE 23606,95:
POKE 23607,248: PRINT AT 10,6;" 
"
5003 FOR f=4 TO 9: PRINT AT f,0;
 OVER 1; INK 6;"       ": NEXT f
: PRINT OVER 1; INK 6;AT 10,1;"
 ";AT 10,5;"  "
5005 PRINT AT 10,0; INK 0; OVER
1;"  ";AT 10,3;"  ": POKE 23606,0
: POKE 23607,60
5010 PAUSE 1: PAUSE 50
5015 PRINT AT 5,25; OVER 1; INK
6;"  ";AT 6,25;"  ";AT 7,25;"  "
;AT 9,25;"  ";AT 10,25;"  ";AT 1
1,25;"  ";AT 13,24;"  ";AT 14,2
4;"  ";AT 15,24;"  "
5020 PRINT AT 3,9; OVER 1; INK 6
;"  ";AT 4,9;"  ";AT 5,9;"  ";AT
 8,8;"  ";AT 9,8;"  ";AT 10,8;"
 ";AT 12,8;"  ";AT 13,8;"  ";AT
14,8;"  "
5025 PRINT AT 10,0; INK 6; OVER
1;"  ";AT 10,3;"  ": PRINT AT 13,
5; INK 0; OVER 1;"  ";AT 14,5;"
 ": POKE 23606,119: POKE 23607
,251: PRINT AT 10,26;"mn";AT 11,
25;",!("
5027 FOR f=1 TO 10: PRINT INK 0;
 OVER 1;AT 10,26;"m";AT 13,6;"/"
: PAUSE 20: NEXT f
5030 POKE 23606,0: POKE 23607,60
: PRINT AT 13,5; INK 6; OVER 1;"
 ";AT 14,5;"  ": POKE 23606,1
19: POKE 23607,251: PRINT INK 6;
AT 10,25;"0": POKE 23607,254: P
RINT INK 6;"  ": PRINT INK 6;AT 9
,25;"(": POKE 23607,251: PRINT
INK 6;AT 11,25;")"
5031 POKE 23606,0: POKE 23607,60
: PRINT AT 14,1; INK 0; OVER 1;"
  ";AT 15,1;"  ";AT 16,1;"
 ";AT 9,28;"  ";AT 10,28;"  "
;AT 11,28;"  ": POKE 23606,119
: POKE 23607,251: PRINT AT 9,27;"
c";AT 10,27;"a";AT 11,27;"  "
5034 FOR f=1 TO 10: POKE 23606,0
: POKE 23607,60: PRINT INK 6; OV
ER 1;AT 13,1;"  ";AT 8,29; INK 6
;"  ": PAUSE 5: PRINT INK 0; OVE
R 1;AT 13,1;"  ";AT 8,29; INK 0;
"  "
5037 PAUSE 5: POKE 23606,119: PO
KE 23607,251: PRINT AT 15,2;"K";
AT 10,29;"O": PAUSE 5: PRINT AT
15,2;"H";AT 10,29;"L": PAUSE 5:
NEXT f
5040 POKE 23606,0: POKE 23607,60
: FOR f=13 TO 16: PRINT OVER 1;
 INK 6;AT f,1;"  ";AT f-5,27;"
 ": NEXT f: GO TO 67
5100 PAUSE 1: IF c=12 THEN FOR f
=1 TO 3: PRINT AT 12,10; OVER 1;
 INK 2;"  ";AT 13,10;"  ": BEEP
.1,-20: PAUSE 20: PRINT OVER 1;
INK 6;AT 12,10;"  ";AT 13,10;"
": NEXT f: GO TO 5103
5101 IF c=8 THEN FOR f=1 TO 3: P
RINT AT 8,10; OVER 1; INK 2;"
";AT 9,10;"  ": BEEP .1,-20: PAUS
E 20: PRINT OVER 1; INK 6;AT 8,1
0;"  ";AT 9,10;"  ": NEXT f: GO
TO 5103
5102 IF c=3 THEN FOR f=1 TO 3: P
RINT AT 3,11; OVER 1; INK 2;"
";AT 4,11;"  ": BEEP .1,-20: PAUS
E 20: PRINT OVER 1; INK 6;AT 3,1
1;"  ";AT 4,11;"  ": NEXT f: GO
TO 5103
5103 PRINT AT 16,10; INK 2; OVER
 1;"  ";AT 17,10;"  ": BEEP .3,-
20
5104 PRINT OVER 1; INK 6;AT 16,1
0;"  ";AT 17,10;"  "
5126 POKE 23606,0: POKE 23607,60
: PRINT AT 14,1; INK 0; OVER 1;"
  ";AT 15,1;"  ";AT 16,1;"
 "
5127 PRINT AT 3,9; OVER 1; INK 6
;"  ";AT 4,9;"  ";AT 5,9;"  ";AT
 8,8;"  ";AT 9,8;"  ";AT 10,8;"
 ";AT 12,8;"  ";AT 13,8;"  ";AT
14,8;"  "
5128 FOR f=1 TO 3: PAUSE 7: POKE
 23606,0: POKE 23607,60: PRINT I
NK 6; OVER 1;AT 13,1;"  ": PAUSE
 7: PRINT INK 0; OVER 1;AT 13,1;
"  ": PAUSE 7
5129 POKE 23606,119: POKE 23607,
251: PRINT AT 15,2;"K";AT 14,3;"
UV": BEEP .1,-20: PRINT AT 15,2;
"H";AT 14,3;"(": BEEP .1,-20: N
EXT f
5130 GO SUB 3100: GO SUB 5300: G
O TO 5040
5200 PAUSE 1: IF c=6 THEN FOR f=
1 TO 3: PRINT AT 6,23; OVER 1; I
NK 2;"  ";AT 7,23;"  ": BEEP .1,
-20: PAUSE 20: PRINT OVER 1; INK
 6;AT 6,23;"  ";AT 7,23;"  ": NE
XT f: GO TO 5203
5201 IF c=10 THEN FOR f=1 TO 3:
PRINT AT 10,23; OVER 1; INK 2;"
 ";AT 11,23;"  ": BEEP .1,-20: P
AUSE 20: PRINT OVER 1; INK 6;AT
10,23;"  ";AT 11,23;"  ": NEXT f
5203 PRINT AT 16,22; INK 2; OVER
 1;"  ";AT 17,22;"  ": BEEP .3,-
20: PRINT AT 16,22; OVER 1; INK
6;"  ";AT 17,22;"  "
5210 GO TO 5226
5215 FOR f=1 TO 3: PRINT AT 13,2
7; INK 2; OVER 1;"  ": BEEP .1,-
20: PAUSE 1: PAUSE 15: PRINT INK
 6; OVER 1;AT 13,27;"  ": NEXT f
5217 PRINT AT 16,26; OVER 1; INK
 2;"  ";AT 17,26;"  ": BEEP .3,-
20: PRINT OVER 1; INK 6;AT 16,26
;"  ";AT 17,26;"  "
5226 PRINT AT 5,25; OVER 1; INK
6;"  ";AT 6,25;"  ";AT 7,25;"  "
;AT 9,25;"  ";AT 10,25;"  ";AT 1
1,25;"  ";AT 13,24;"  ";AT 14,2
4;"  ";AT 15,24;"  "
5227 POKE 23606,0: POKE 23607,60
: PRINT OVER 1; INK 0;AT 9,28;"
 ";AT 10,28;"  ";AT 11,28;"
": POKE 23606,119: POKE 23607,25
1: PRINT AT 9,27;"c";AT 10,27;"a
";AT 11,27;"  "
5228 FOR f=1 TO 3: PAUSE 7: POKE
 23606,0: POKE 23607,60: PRINT I
NK 6; OVER 1;AT 8,29; INK 6;"  "
: PAUSE 7: PRINT INK 0; OVER 1;A
T 8,29; INK 0;"  "
5229 PAUSE 7: POKE 23606,119: PO
KE 23607,251: PRINT AT 10,29;"O"
;AT 9,27;"]!": BEEP .1,-20: PRIN
T AT 10,29;"L";AT 9,27;"ca": BEE
P .1,-20: NEXT f
5230 GO SUB 3100: GO SUB 5300: G
O TO 5040
5300 IF c$=1 THEN LET c$=0
5301 POKE 23606,0: POKE 23607,60
: PRINT OVER 1; INK 6;AT 3,20;"
 ": LET mi=mi+1: PRINT OVER
1; INK 0;AT 3,20;("  " AND mi>0
); INK 6;"  ": FOR f=1 TO mi: PR
INT OVER 1; INK 0;"  ";: NEXT f:
IF mi=3 THEN GO TO 5310
5305 RETURN
5310 LET a$="   DESPEDIDO DE LA
 FABRICA   "
5315 PRINT INK 7; PAPER 7;AT 20,
0;a$: FOR f=1 TO LEN a$
5317 POKE 63784,0: POKE 637..,(c
am+4 AND cam<63): POKE 63786,50
: LET a=INT (RND*6)
5318 PRINT AT 20,f-1; INK a;a$(f
): NEXT f
5320 IF INKEY$="" THEN RANDOMIZE
 USR 63781: GO TO 5320
5330 IF p>rec THEN LET rec=p: LE
T rec2=cam
5340 POKE 63784,126: POKE 63788,
6: POKE 63792,2: GO TO 2
6000 GO SUB 6070: BORDER 7: INK
0: PAPER 7: CLS
6020 PRINT AT 10,2;"PULSA UNA TE
CLA PARA EMPEZAR"
6060 PAUSE 1: PAUSE 0: RUN
6070 RESTORE 6080: FOR f=63781 T
O 63801: READ a: POKE f,a: NEXT
f: RETURN
6080 DATA 33,0,64,126,211,254,62
,6,211,254,62,2,211,254,35,124,2
54,91,32,239,201
9997 PAPER 0: BORDER 0: INK 7: C
LEAR 63780: PRINT AT 12,0; FLASH
 1;"ESPERA UN MOMENTO CARGANDO E
L CM": LOAD ""CODE : GO TO 6000
9998 SAVE "MARIO BROS" LINE 9997
9999 POKE 23606,0: POKE 23607,60
: REM DEVUELVEN LOS CARACTERES
       ORIGINALES DEL SPECTRUM
```



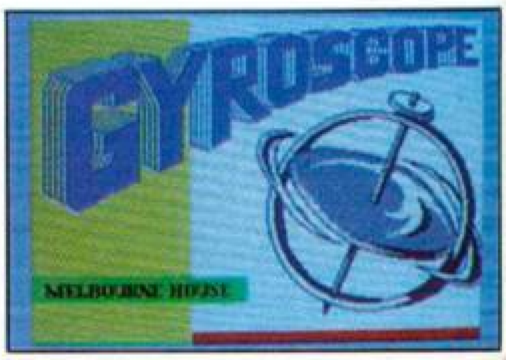

128K

Gyroscope • Melbourne House • Arcade

LA PEONZA HABILIDOSA



Algunos de vosotros habréis tenido la oportunidad de ver un juego muy parecido a éste en las máquinas de los billares. Ahora, Melbourne House ha realizado una versión llamada Gyroscope, para que puedas disfrutarlo en tu ordenador.

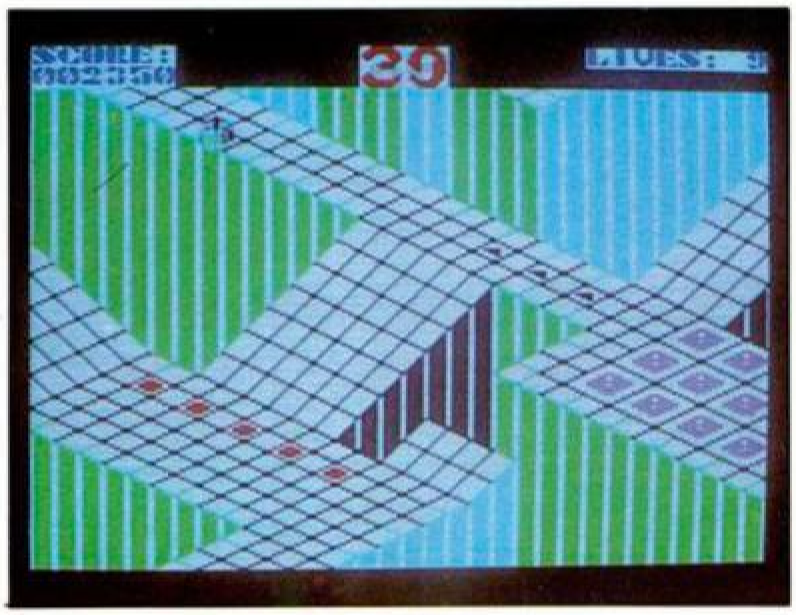


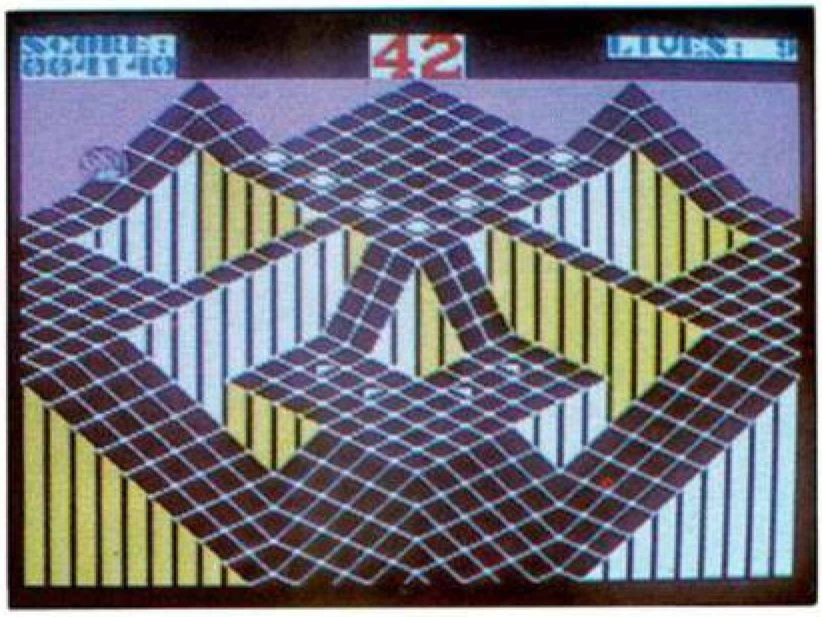
Sobre este juego no es posible realizar ningún tipo de comentario relacionado con la historia que le rodea. Gyroscope no trata de rescates de princesas, ni de héroes que deben enfrentarse a malvados enemigos, ni siquiera trata de simular el funcionamiento de ninguna nave especial. Este juego consiste, lisa y llanamente, en llevar un giróscopo (peonza, para entendernos mejor), a través de un complicado circuito.

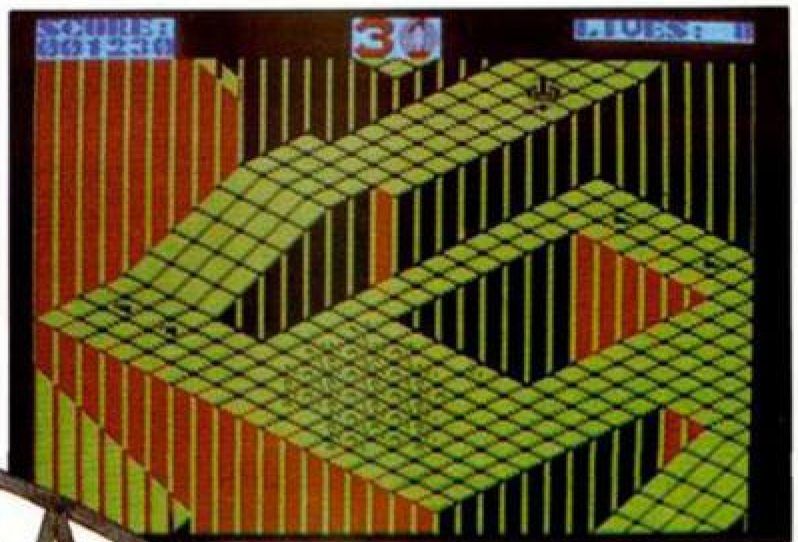
Por el contrario, y a pesar de la simplicidad de su planteamiento, el desarrollo de el juego resulta bastante complicado.

Al principio los primeros laberintos por los que debe moverse la peonza son relativamente fáciles, pero en cuanto conseguimos pasar las primeras tres o cuatro pantallas, las cosas cambian notablemente.

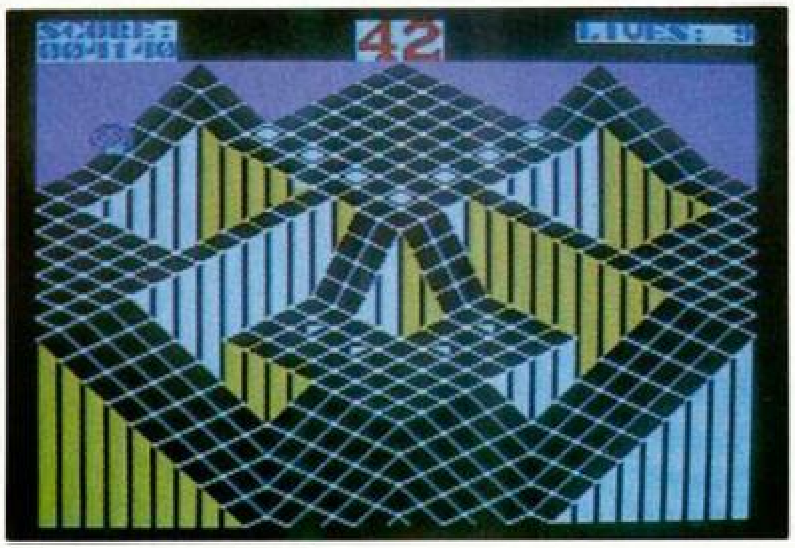
Los pasadizos se hacen cada vez más intrincados, más estrechos, y todo esto unido a la inercia de la propia peonza (que hace que ya de por sí resulte bastante complicado hacerse con el dominio de la misma), provoca que en algunos momentos parezca casi imposible poder completar el recorrido. Si a esto le añadimos algún que otro obstáculo como bolas, trampas en el suelo, zonas deslizantes, etc., el resultado, como os podréis imaginar, es en algunos casos, alarmante.



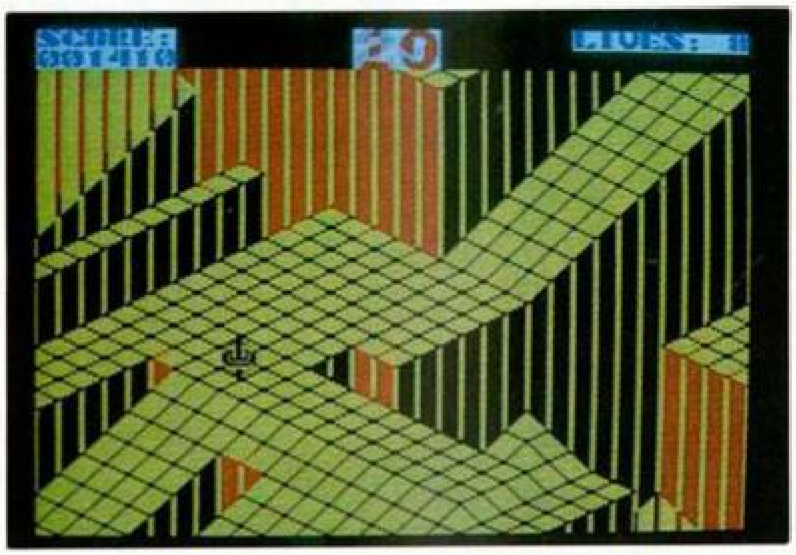












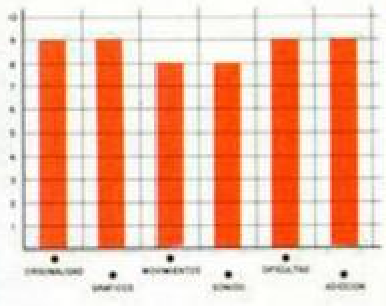


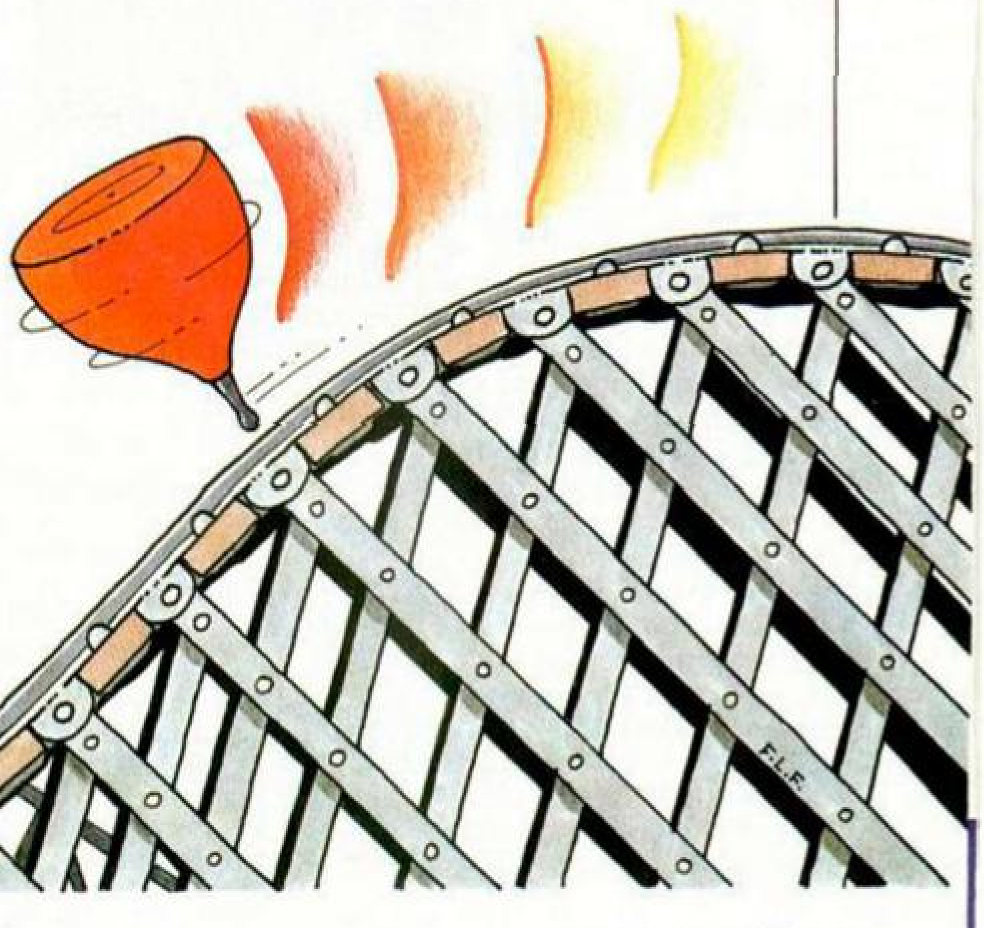
Otro pequeño inconveniente es que el número de vidas de las que se dispone, a pesar de ser bastante elevado para lo que estamos acostumbrados —seis—, resulta desde todo punto insuficiente no ya sólo para llegar hasta el final del juego, sino para conseguir pasar un número aceptable de pantallas sin tener la necesidad de volver a comenzar desde la primera.

Sin embargo, a pesar de todas estas dificultades, y quizá precisamente por ellas, Gyroscope es un juego realmente entretenido y muy emocionante, a pesar de que en algunos momentos nos gustaría que las cosas fueran un poco más sencillas.

Otro punto destacable de este juego son sus gráficos, con los que se ha conseguido darle un verdadero efecto de tridimensionalidad, además de resultar brillantes en cuanto a formas y colorido.

En definitiva, es un juego original, bien realizado y bastante adictivo.

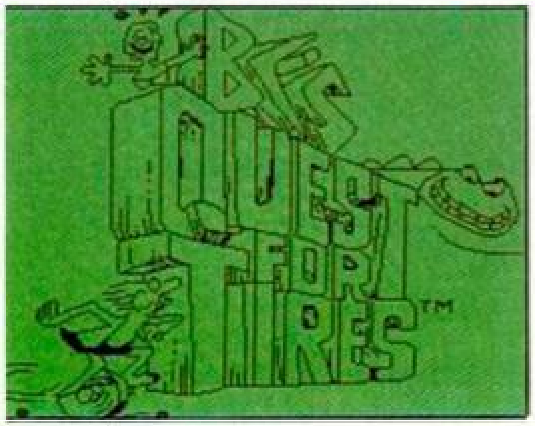






BC'S Quest for Tires • Software • Arcade

EL RAPTO DEL DRAGON



De este juego se puede decir que no ha sido realizado con demasiadas aspiraciones, pero en definitiva resulta entretenido y te puedes pasar un buen rato intentando que Thor consiga rescatar a su único amor,



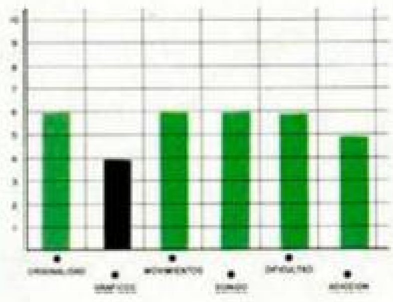
Cute Chick, de las garras de el malvado dragón que la tiene prisionera.

Para ello deberemos de sortear, montados en nuestro pétreo monociclo, los numerosos baches y cascotes que se encuentran repartidos por estas carreteras prehistóricas. Además de estos inconvenientes, existen muchos otros como árboles, ríos y troncos, que dificultarán enormemente el que consigamos llegar hasta los brazos de Cute Chick.

Sin embargo, si el evitar estos obstáculos te resulta sencillo, tienes la posibilidad de elegir entre cuatro niveles diferentes de dificultad, al igual que podrás determinar el número de jugadores (1 o 2).

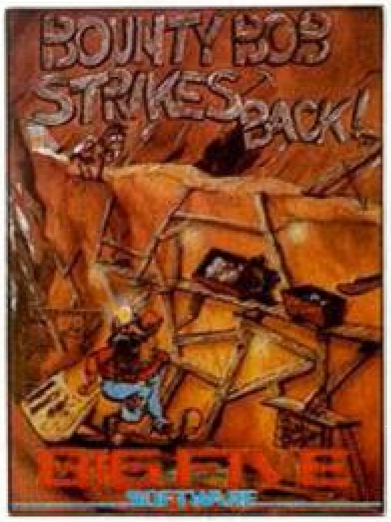
Los gráficos tienen una apariencia, aunque simpatica, un tanto pobre y simplona; sin embargo, el scroll de la pantalla está realizado con mucha suavidad, por lo que se le imprime al juego una sensación muy buena de movimiento, no sólo del personaje, sino también del paisaje de fondo.

Una de las caracteristicas más curiosas de este BC'S, es que al cargarse el programa y con el fin de evitar las copias piratas, aparece en la pantalla un mensaje que te dice que teclees un código de colores, (que varía cada vez que es cargado) y que viene escrito en una lámina que se adquiere junto con la copia. Tienes dos oportunidades para teclear el código correcto, de lo contrario el programa se autodestruye.



Bounty Bob • U.S. Gold • Arcade

EN LAS PROFUNDIDADES DE LA MINA



Bounty Bob vuelve al ataque en una nueva y complicada aventura, continuación de una anterior llamada Miner 2049er.

El escenario en el que se desarrolla el juego es bastante similar al anterior aunque, en esta ocasión, los túneles y pasadizos de la mina son algo más tortuosos por lo que resulta muy complicado moverse sin cometer



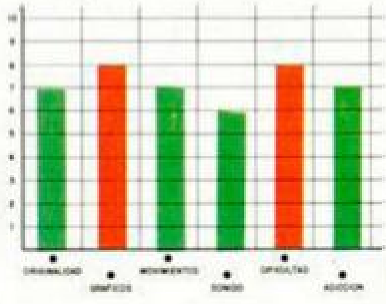
ningún resbalón y acabar espanzurrado contra alguna roca mal situada o contra el mismo suelo de la mina.

La principal tarea de Bounty consiste en recorrer todas las partes de la estructura de la mina. Para ello tan sólo debe limitarse a pasar por encima de ellos, pero claro, las cosas en la práctica no son tan sencillas con en la teoría.

La dificultad se encuentra, además de en tratar de evitar a las peligrosas plagas de mutantes que se encuentran en cada una de las salas de la mina, en conseguir que los saltos de una sección a otra sean precisos y lograr enviar a Bounty al lugar exacto deseado, de lo contrario, si la caída es larga, se matará.

Además, por toda la mina se encuentran repartidos una serie de elementos que te ayudarán en gran medida a moverte por las diferentes salas: ascensores, elevadores hidráulicos, tubos de succión, transportadores, carretillas,... aunque también existen otro tipo de objetos que te pueden servir para inmunizarte de los organismos mutantes: tesoros y barras de super energía.

El final del juego se logra cuando las 25 salas de la intrincada mina quedan completadas. Con ello conseguimos destruir los enrevesados planes del malvado Yukón Yohan.



RUTINA PARA DESCOMPRIMIR PANTALLAS

Miguel SEPULVEDA

Hace un par de números comentamos en esta misma sección una rutina que comprimía los archivos de presentación visual con objeto de conseguir almacenar en memoria un mayor número de pantallas. Esta rutina no tendría gran utilidad si no es complementada con otra que ahora os presentamos y que permite hacer lo contrario, es decir, recuperar las pantallas desde la memoria.

En primer lugar vamos a intentar comprender la filosofía de la rutina con el mismo ejemplo que pusimos en el artículo anterior.

A partir de los archivos de presentación visual y de atributos, habíamos hecho una reducción de bytes consistente en agrupar todos aquéllos que se repetían y eran consecutivos. Habíamos obtenido un par de contadores y una serie de valores que guardábamos en memoria, de esta forma conseguíamos almacenar un gran número de pantallas. El objeto del presente artículo, es recomponer los archivos iniciales, y poder de esta forma representarlos en pantalla.

De una sucesión de bytes cuyos valores eran:

Ø,Ø,1,1,1,1,2,3,5,5,5,1Ø,1Ø,1Ø,1Ø,1Ø,

1Ø,1Ø,1Ø,2Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,Ø,

1,1,1,1,1,1,1,1

obteníamos estos valores:

Ø,2,Ø,Ø,8Ø,4,1,Ø,5,2,3,5,5,5,8Ø,8,

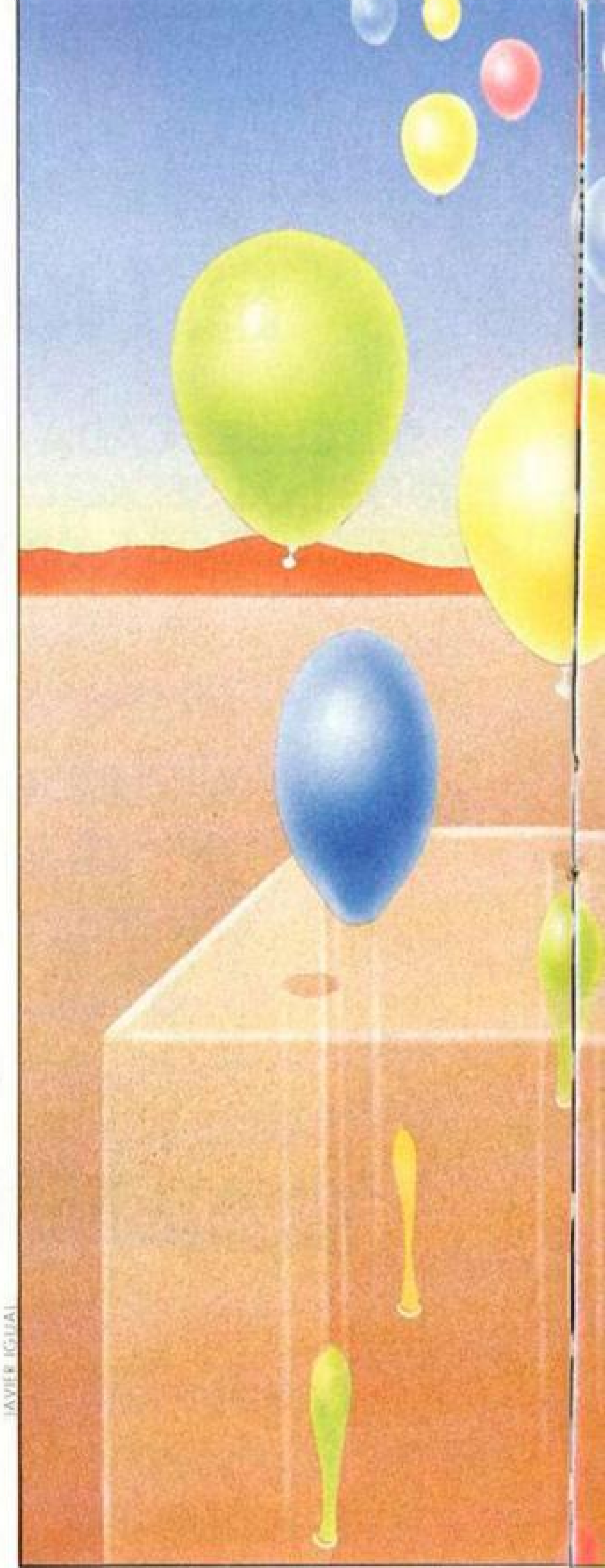
1Ø,Ø,1,2Ø,8Ø,1Ø,Ø,8,7,1

que eran los que almacenábamos en memoria. El proceso actual será partir de esta sucesión de valores para obtener los iniciales.

Si empezamos a leer esta sucesión, encontramos el valor de un contador (Ø,2) que nos indica, por un lado (el Ø) que se trata de un contador de bytes desiguales y por otro (el 2) que a continuación vienen dos bytes cuyos valores son los dos siguientes números de la sucesión (el Ø y el Ø), entonces, estos dos últimos valores son los que tomaríamos como pertenecientes al archivo de presentación visual y de atributos.

A continuación seguimos leyendo y encontramos otro contador (8Ø,4) que nos indica que se trata de un contador de bytes iguales (por el 8Ø) y que a continuación vienen 4 bytes iguales de valor 1 (el siguiente número de la sucesión), entonces lo que tendríamos que hacer sería colocar cuantro bytes a 1 a continuación de los dos bytes a cero que habíamos puesto anteriormente.

Siguiendo este procedimiento hasta el final obtendríamos la serie de bytes que había antes de la «comprensión».

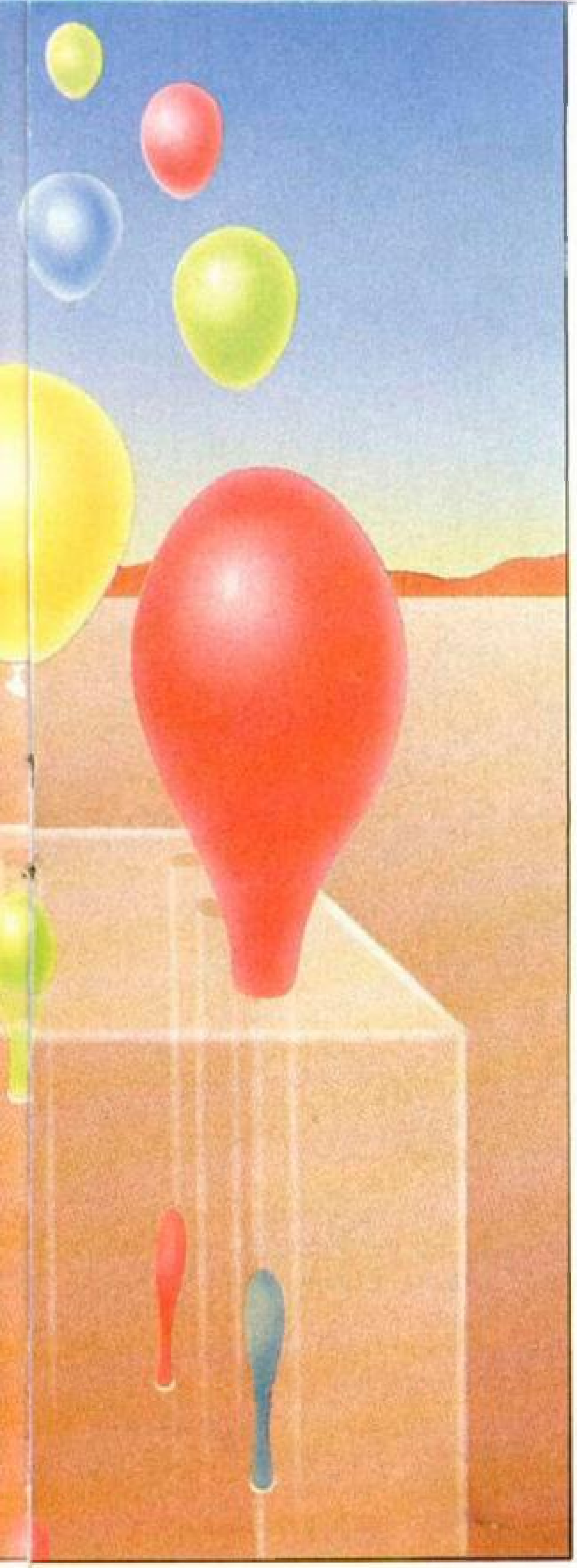


Funcionamiento de la rutina

Pasemos a ver cómo trabaja esta rutina. Lo primero que hace es cargar el registro HL con la dirección de memoria donde empieza el archivo comprimido, y el registro DE con la dirección del principio del archivo de presentación visual. Entre las líneas 8Ø y 2ØØ del listado, la rutina carga en el acumulador el byte más significativo del contador de bytes diferentes o de bytes iguales y comprueba si es el código de fin de archivo, si es afirmativo es que ha expandido ya el archivo y entonces sale de la rutina; en caso contrario carga en el registro BC el contador de bytes a procesar y pregunta si están comprimidos, y si no lo están, pasa todos los bytes que indica el registro BC a los archivos de presentación visual y de atributos.

La rutina averigua si los bytes están comprimidos o no, preguntando por el bit más significativo del contador, si es 1 es que están comprimidos, ya que en la rutina de comprimir, cuando los bytes estaban repetidos más de tres veces seguidas, al contador se le sumaba 8ØØØH.

Entre la línea 22Ø del listado y el final de la rutina, se tratan los bytes comprimidos. Lo primero que hace es borrar el bit más significativo del contador (que estaba a 1). A continuación, toma del archivo comprimido el byte que hay que expandir, carga en el registro HL la dirección de los archivos de imagen después de haber salvado su contenido en el stack, pone el byte a expandir en los archivos de imagen y lo



extiende por los archivos tantas veces como indica el registro BC que hace de contador.

Esta rutina es para ser usada desde el BASIC. Si quieres llamarla desde otra rutina en código máquina hay que añadir las siguientes instrucciones: la línea 5Ø del listado anterior quedará suprimida.

CAMBIOS NECESARIOS PARA LLAMAR A LA RUTINA DESDE CODIGO MAQUINA

```
41         DEC  A          51 PMP   EQU  $
43         JR   Z,PMP      53       LD   B,(HL)
45         INC  HL         55       INC  HL
47         INC  HL         57       LD   L,(HL)
49         JR   START      59       LD   H,B
50 ;
```

LISTADO ASEMBLER DE LA RUTINA DESCOMPENSADORA

```
10 ;
20          ORG  32768
30 ;
40 PGSMP    EQU  $
50          LD   HL,(23670)
60          LD   DE,#4000
70 ;
80 PMP0     EQU  $
90          LD   A,(HL)
100         CP   #FF
110         RET  Z
120         LD   B,A
130         INC  HL
140         LD   C,(HL)
150         INC  HL
160         SLA  B
170         JR   C,PMP2
180         SRL  B
190         LDIR
200         JR   PMP0
210 ;
220 PMP2    EQU  $
230         SRL  B
240         LD   A,(HL)
250         INC  HL
260         PUSH HL
270         LD   H,D
280         LD   L,E
290         INC  DE
300         LD   (HL),A
310         LDIR
320         POP  HL
330         JR   PMP0
340         END
```

Estas instrucciones sólo son válidas si se ha creado una tabla con las direcciones de los archivos comprimidos y se llama a la rutina con el número del archivo a expandir en el acumulador (empezando por 1) y la dirección de la tabla de direcciones en HL. La explicación de las instrucciones agregadas es

LISTADO 1

```
1  2A765C1100407EFEFFC8 1168
2  47234E23CB203806CB38 775
3  EDB018EECB387E23E562 1422
4  6B1377EDB0E118E00000 1131
```

la siguiente: al ocupar las direcciones de memoria dos bytes, y estando las direcciones de los archivos comprimidos puestas en la tabla consecutivamente, entonces se va incrementando el índice de la tabla que es el registro HL dos bytes tantas veces como indica el acumulador, que es el número de archivo pedido y la dirección efectiva de dicho archivo se carga en el registro HL. A continuación, empieza la parte de la rutina que hace la expansión.

Cuando se llama desde el BASIC, se usa la rutina sin modificar, y si existe la tabla de direcciones, se calcula la dirección del archivo comprimido que queremos en el BASIC y se pasa a la variable del sistema SEED, llamándola de la siguiente forma:

RANDOMIZE x: RANDOMIZE USR dirección en memoria de la rutina

Donde x, es una variable con la dirección del archivo comprimido o el valor absoluto de la dirección.

También se puede utilizar:

POKE 23670,	byte bajo de la dirección del archivo comprimido
POKE 23671	byte alto de la dirección del archivo comprimido
RANDOMIZE USR	dirección en memoria de la rutina

Utilización

La rutina de código máquina puede teclearse usando un ensamblador o con el Cargador Universal de Código Máquina publicado en el número 31 (utilizando el listado 1). El DUMP se hará en la dirección 32768 y se salvará en cinta con el nombre «expansor» indicando 32768 como dirección y 38 como número de bytes. Para cargar luego la rutina, al ser reubicable, se hace de la siguiente forma:

LOAD " " CODE direc. de memoria, 38

El PROGRAMA 1 usa la rutina en código máquina del primer artículo para crear un bloque de archivos de imagen comprimidos. Lo primero que hace es pedir donde se alamcenarán en memoria los archivos comprimidos (la dirección más baja es la 26ØØØ), y los archivos que se van a comprimir.

A partir de la dirección de memoria que se ha dado, reserva dos bytes por cada pantalla que se desee comprimir, más dos bytes. El primero para poner el número de archivos comprimidos que tiene el bloque, y el último lo pone a cero para indicar que es el último byte de la tabla de direcciones de los archivos comprimidos; a partir de ese byte se almacena el primer archivo comprimido. A continuación, carga la rutina de comprimir los archivos de imagen en el buffer de impresora en la dirección 233ØØ, y te informa para que pongas la cinta con las pantallas a comprimir; a continuación, el programa inicia un bucle que se ejecuta tantas veces como pantallas haya que comprimir y va rellenando la tabla con la dirección de

memoria donde se almacena cada uno de los archivos comprimidos.

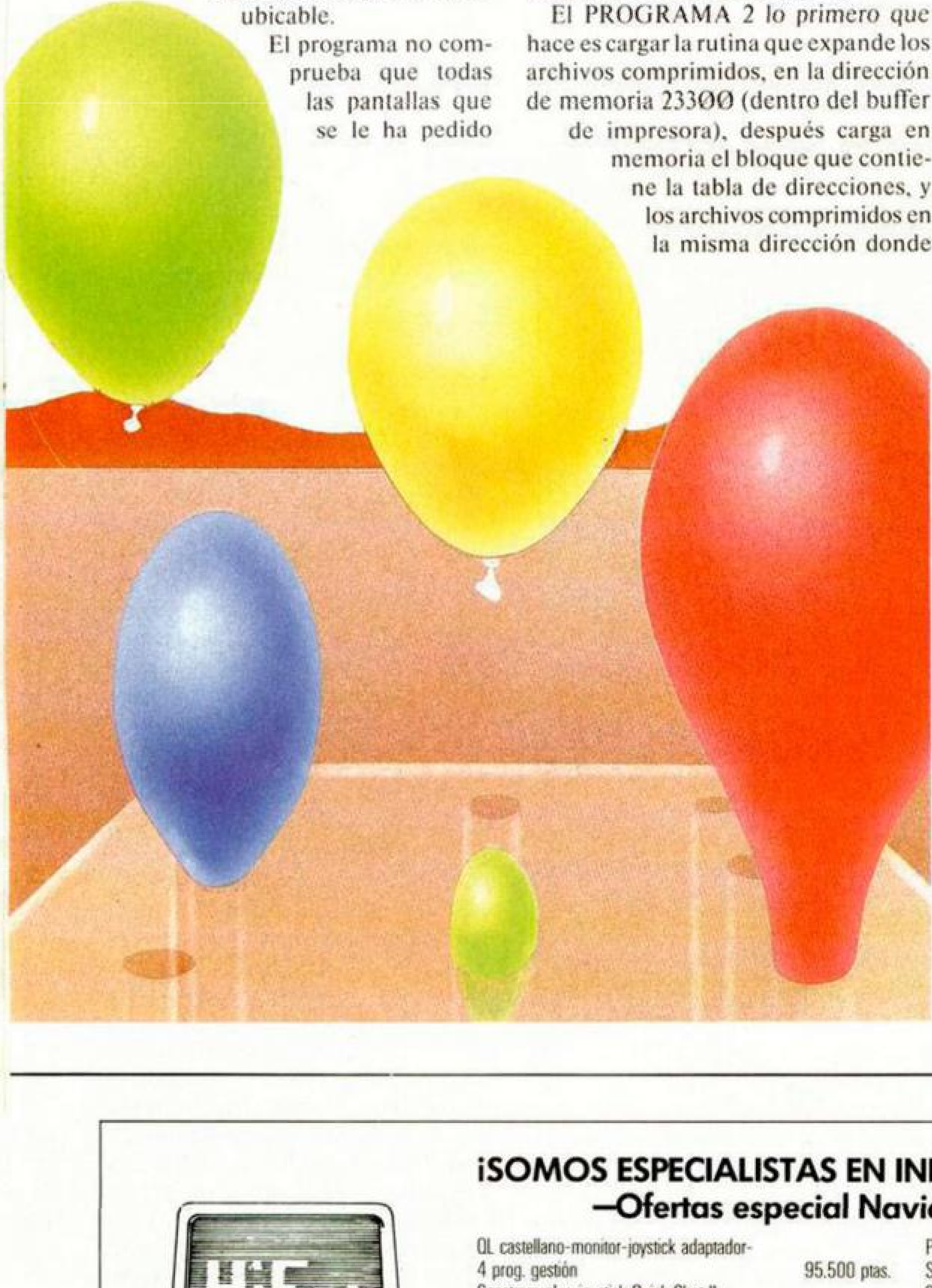
Seguidamente informa de los bytes que ocupa cada archivo comprimido y pide el nombre con que se quiere salvar el bloque compuesto por la tabla de direcciones y los archivos comprimidos. Este bloque no es reubicable.

El programa no comprueba que todas las pantallas que se le ha pedido que comprima quepan en la memoria, ya que de antemano ignora los bytes que va a ocupar cada archivo comprimido.

El sistema dará un error si se sobrepasa la memoria libre. En este caso, el usuario deberá ir eliminando pantallas hasta que entren en memoria.

El PROGRAMA 2 *lo primero que* hace es cargar la rutina que expande los archivos comprimidos, en la dirección de memoria 233ØØ (dentro del buffer de impresora), después carga en memoria el bloque que contiene la tabla de direcciones, y los archivos comprimidos en la misma dirección donde se creó (si se quiere cambiar la dirección de carga en memoria, se deberá hacer antes de expandir los archivos, corrigiendo las direcciones de la tabla según el desplazamiento respecto a su dirección de origen).

Después entra en un bucle que va expandiendo y mostrando los archivos comprimidos en la pantalla. En esta demostración, el tiempo que está expuesta una pantalla, depende del PAUSE de la línea 8Ø.



PROGRAMA 1 COMPRESOR DE PANTALLAS

```
  10 CLEAR 25999: INPUT "Direcci
on donde se almacena los archivo
s comprimidos= ";m
  20 INPUT "Num. de pantallas a
comprimir ";n
  30 DIM z(20): LET dm=m+n+2+2:
POKE m,n: LET i=m: LET m=m+1
  40 LOAD ""CODE 23300,116: REM
carga la rutina de comprimir
  50 CLS : PRINT AT 12,5;"PONGA
CINTA CON SCREENS";AT 14,5;"A CO
MPRIMIR Y PULSE EL";AT 16,5;"PLA
Y DEL CASSETE"
  60 FOR x=1 TO n
  70 LOAD ""SCREEN$
  80 RANDOMIZE dm: POKE m,PEEK 2
3671: POKE m+1,PEEK 23670: LET m
=m+2: LET l=USR 23300
  90 LET dm=dm+l: LET z(x)=l
 100 NEXT x
 110 POKE m,0: LET dm=dm-i
 120 CLS : PRINT FLASH 1;AT 1,5;

"LONGITUD DE LOS SCREENS"
 130 FOR x=1 TO n
 140 PRINT BRIGHT 1; PAPER 2; IN
K 6;AT x+3,4;"SCREEN ";x;" = ";z
(x);" BYTES"
 150 NEXT x
 160 DIM w$(10): INPUT "Nombre c
on el que quiere salvar el bloqu
e de screens= "; LINE w$
 170 SAVE w$CODE i,dm
 180 STOP
```

PROGRAMA 2 EXPANSOR DE PANTALLAS

```
  10 CLEAR 25999: PRINT AT 11,3;
"CARGA LA RUTINA DE EXPANDIR": L
OAD ""CODE 23300,38
  20 INPUT "Direccion donde se h
a cargado el bloque= ";m: LET n=
PEEK m: LET m=m+1
  30 CLS : PRINT AT 11,7;"CARGA
EL BLOQUE DE";AT 13,6;"ARCHIVOS
COMPRIMIDOS": LOAD ""CODE
  50 FOR x=1 TO n
  60 LET d=256*PEEK m+PEEK (m+1)
: LET m=m+2
  70 RANDOMIZE d: LET l=USR 2330
0
  80 PAUSE 10
  90 NEXT x
 110 STOP
```

PROGRAMA 1

```
  10 CLEAR 30999
  20 FOR n=31000 TO 31010
  30 READ a: POKE n,a: NEXT n
  40 DATA 58,177,92,71,58,176,92
,128,245,193,201
  50 INPUT "Primer operando ? ";
a
  60 IF a>255 OR a<0 THEN GO TO
50
  70 POKE 23728,a
  80 INPUT "Segundo operando ? "
;a
  90 IF a>255 OR a<0 THEN GO TO
80
 100 POKE 23729,a
 110 LET a=USR 31000
 120 GO SUB 3100
 130 FOR n=LEN a$ TO 15
 140 LET a$="0"+a$: NEXT n
 200 CLS : PRINT '''
 210 PRINT TAB 3;"ACUMULADOR","
   SZxHxVNC"
 220 PRINT '" A= ";a$(1 TO 8),"
F= ";a$(9 TO 16)
 230 GO TO 50
3100 REM DEC. (a) A BIN. (a$)
3110 LET a$="": LET c=a
3120 LET coc=INT (c/2): LET res=
c-coc*2: LET e$=STR$ INT res: LE
T a$=e$+a$: LET c=coc: IF c>=2 T
HEN GO TO 3120
3130 LET e$=STR$ INT c: LET a$=e
$+a$: RETURN
```

La operación ya se ha realizado, tenemos el resultado en "BC" y, por tanto, será lo que obtengamos al retornar a Basic. Ahora sólo nos falta sacar, de alguna forma, el contenido del registro "F" (indicadores) de forma que lo podamos leer desde Basic. Para ello, las líneas 120 y 130 pasan los contenidos de "A" y "F" a "BC" y la línea 140 almacena el contenido de "E" en la posición de memoria 5CB0h (23728), desde donde será leído por el Basic. En esta operación, también se guarda en 23729 el contenido del registro "D" pero, en este caso, no nos interesa.

Sería intenresante que el lector intentara, ahora, ensamblar por su cuenta este programa, para ello, deberá proceder como hicimos nosotros en el caso anterior. Primero, copie el programa en un papel, ahora, vaya buscando cada instrucción en las tablas ("AND A" se ensambla como A7h ó 167d). A continuación, escriba los operandos numéricos sin olvidar invertir el orden de los octetos y, finalmente, acuérdese de ensamblar "RET" como C9h ó 201d.

¿Ya lo tiene? Correcto, ahora compruebe si lo que usted ha ensamblado coincide con lo nuestro:

Assembler	Hexadecimal	Decimal
AND A	: A7	: 167
LD A,(#5C76)	: 3A,76,5C	: 58,118,92
LD D,A	: 57	: 87
LD A,(#5CB0)	: 3A,B0,5C	: 58,176,92
ADC A,D	: 8A	: 138
LD C,A	: 4F	: 79
LD A,(#5C77)	: 3A,77,5C	: 58,119,92
LD D,A	: 57	: 87
LD A,(#5CB1)	: 3A,B1,5C	: 58,177,92
ADC A,D	: 8A	: 138
LD B,A	: 47	: 71
PUSH AF	: F5	: 245
POP DE	: D1	: 209
LD (#5CB0),DE	: ED,53,B0,5C	: 237,83,176,92
RET	: C9	: 201

No se preocupe si se ha equivocado en algo, sería mucho pedir que el primer programa que ensambla le saliera sin errores. Ahora, con los datos de la tercera columna (donde dice: "Decimal") podemos construir el programa en Basic que introduzca esta rutina en memoria, y la utilice para sumar dos números. Este programa es el listado que aparece con el nombre de PROGRAMA 2. No hace falta que lo copie entero, si lo desea, puede cargar el PROGRAMA 1 y reescribir las líneas 20, 40, 60, 70, 90, 100, 120, 130, 210 y 220 ya que son las únicas que varían.

El funcionamiento es muy similar al del PROGRAMA 1, pero observe que en las líneas 70 y 100 partimos los operandos en dos octetos antes de meterlos en sus direcciones de memoria correspondiente.

Pruebe con distintos operandos y verá que el programa suma correctamente; siempre que el resultado sea mayor de 65535, obtendrá ese resultado menos 65536 y el indicador de acarreo se pondrá a "1".

Puede también restar números si cambia "ADC A,D" por "SBC A,D"; para ello, puede cambiar los dos "138" de la línea 40 (del Basic) por "154", o bien, detener el programa (con STOP) y teclear: POKE 31008,154: POKE 31017,154. Luego, no lo arranque con "RUN", sino, con "GO TO 50".

Como verá, el hecho de POKEar en la zona de programa hace que éste se comporte de forma distinta; he aquí la razón de los famosos "POKES" de "vidas infinitas" y similares que se utilizan en juegos comerciales (sección "Patas Arriba" de la revista MICROMANIA).

En una resta convencional, operando con números decimales, cuando el valor del sustraendo es mayor que el minuendo en la unidad enfrentada; sumamos diez al valor del minuendo, restamos y "nos llevamos una" para la siguiente unidad, esto es, restamos el diez queu habíamos sumado al minuendo:

```
5724-3615
```

Al resta 4–5 en realidad se hace 14–5 y nos llevamos una (acarreo), este acarreo lo sumamos a la decena 1 del sustraendo, que es lo mismo que restarlo en el minuendo a la decena 2 y resulta: 2–2 ó 1–1.

```
 5724
-3615
    1      acarreo
------
 2109
```

Pues esto mismo ocurre al sumar octetos, cuando el octeto sustraendo es mayor que el octeto minuendo y activarse por tanto el bit de acarreo, al tenerlo en cuenta con los octetos de orden superior; es como si en el minuendo se sumara 256 al octeto inferior y se restara uno al octeto superior.

Ver figura 6-2. Este es el uso más importante de la condición de acarreo para la resta.

```
BINARIO

 01001000   10100011   00101010   MINUENDO
 00100101   00000111   01000001   SUSTRAENDO
0       0          1              ACARREO
 11011011   11111000   10111111   SUSTR. COMPLEMENTADO
 00100011   10011011   11101001   RESULTADO

HEXADECIMAL

   48A32A    MINUENDO
 - 250741    SUSTRAENDO
   0 0 1     ACARREO
   239BE9    RESULTADO

DECIMAL

   72 163  42    MINUENDO
 - 37   7  65    SUSTRAENDO
   0  0   1      ACARREO
   35 155 -23    RESULTADO
```

Fig. 6.2. Resta de varios objetos con acarreo.

SBC A,r

OBJETO:

Resta del registro acumulador "A" el contenido del registro especificado por "r", más el indicador de acarreo. Deja el resultado en el registro acumulador.

CODIGO DE MAQUINA:

10011<—r—<

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 – si el resultado es negativo
pone 0 – en cualquier otro caso
Z pone 1 – si el resultado es cero
pone 0 – en cualquier otro caso
H ; pone 1 – si no hay acarreo desde el bit 3
pone 0 – en cualquier otro caso
N ; pone 1 – siempre
C ; pone 1 – si no hay acarreo desde el bit 7
pone 0 – en cualquier otro caso
P/V; pone 1 – si hay desbordamiento (overflow)
pone 0 – en cualquier otro caso

CICLOS DE MEMORIA:

1

CICLOS DE RELOJ:

4

EJEMPLO:

SUB A,H

Valor del registro "A"

(A): 11111111 FFh

Valor del registro "H"

(H): 01110111 77h

Indicador de acarreo (C) = 0.

Instrucción

SUB A,H 10011100 9Ch

Operación:

```
       (H)   01110111
        +C          0
             --------
             01110111

compl. a 2   10001001
      +(A)   11111111
             --------
             10001000
```

Valor del registro "A" después de la ejecución

(A): 10001000 88h

Indicadores de condición después de la ejecución

S	Z		H		P/V	N	C
1	0	x	0	x	0	1	0

SBC A,n

OBJETO:

Resta del registro acumulador "A" el entero de 8 bits n, más el bit de acarreo. Deja el resultado en el registro acumulador.

CODIGO DE MAQUINA:

11011110	DEh
<——— n ———>	

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 – si el resultado

GRUPO ARITMETICO DE 8 BITS (SUMA Y RESTA)

Código Fuente	Hexadecimal	Decimal
ADD A,A	87	135
ADD A,B	80	128
ADD A,C	81	129
ADD A,D	82	130
ADD A,E	83	131
ADD A,H	84	132
ADD A,L	85	133
ADD A,n	C6,n	198,n
ADD A,(HL)	86	134
ADD A,(IX+d)	DD,86,d	221,134,d
ADD A,(IY+d)	FD,86,d	253,134,d
ADC A,A	8F	143
ADC A,B	88	136
ADC A,C	89	137
ADC A,D	8A	138
ADC A,E	8B	139
ADC A,H	8C	140
ADC A,L	8D	141
ADC A,n	CE,n	206,n
ADC A,(HL)	8E	142
ADC A,(IX+d)	DD,8E,d	221,142,d
ADC A,(IY+d)	FD,8E,d	253,142,d
SUB A	97	151
SUB B	90	144
SUB C	91	145
SUB D	92	146
SUB E	93	147
SUB H	94	148
SUB L	95	149
SUB n	D6,n	214,n
SUB (HL)	96	150
SUB (IX+d)	DD,96,d	221,150,d
SUB (IY+d)	FD,96,d	253,150,d
SBC A,A	9F	159
SBC A,B	98	152
SBC A,C	99	153

samblado el programa antes de mirar la tabla anterior. Prométase a sí mismo que la próxima vez lo intentará.

Ya tenemos preparada la rutina en código máquina para sumar dos números. Seamos buenos con los que aún tienen solo 16K, y carguemos la rutina a partir de la dirección 31000.

Ha llegado el momento de pasar al olvidado Basic. El PROGRAMA 1 se encarga de todo. La línea 10 baja RAMTOP, las líneas 20 y 30 introducen en memoria nuestra rutina que se encuentra en los DATA de la línea 40. Las líneas 50 a 100 nos piden los dos operandos y los POKEan en memoria tras comprobar si están dentro de rango.

La línea 110 llama a nuestra rutina en C/M de forma que, al retornar, el contenido del registro "BC" se almacene en la variable "a". En 120 llamamos a la rutina 3100 que nos pasa el número a binario, esta subrutina es la misma que usábamos en el programa para cambiar de base, del capítulo 3. Las líneas 130 y 140 completan el número con ceros a la izquierda para obtener, de nuevo, 16 bits. Finalmente, las líneas 200 a 220 imprimen en pantalla el valor que contenía el acumulador después de efectuar la suma y el estado de los indicadores en el registro "F". EL significado de los indicadores es el siguiente:

```
S : Signo
Z : Cero
H : Semi-acarreo
V : Desbordamiento
N : Suma/Resta
C : Acarreo
```

Los indicadores marcados "x" presentan un estado indeterminado y no habrá que tomarlos en cuenta.

Una vez que tenga el programa en memoria, pruebe a introducir distintos operandos comprendidos entre 0 y 255. Le sugerimos unos cuantos:

```
 17 +  17 =  34
 15 + 240 = 255 (N)
128 + 128 =   0 (Z,V,C)
127 +   1 = 128 (S,H,V)
  3 + 127 = 130 (S,H,V)
```

Puede utilizar este mismo programa para la resta cambiando "ADD A,B" por "SUB B", es decir, el "128" de la línea 40 por un "144". Haga el cambio y ejecute el programa de nuevo, esta vez restará el segundo operando del primero. Si el segundo es mayor que el primero (resultado negativo), el indicador "C" se pondrá a "1" y el resultado aparecerá en complemento a 2.

Ahora vamos a complicar un poco más la cosa, se trata de hacer una rutina que permita sumar números superiores a 255. En este caso, usaremos la instrucción "ADC" (sumar con acarreo) para poder tener en cuenta, cuando sumemos un octeto, el acarreo procedente del anterior.

Introduciremos el primer operando en las direcciones 5CB0H (23728) y 5CB1h (23729) (primero el octeto menos significativo y luego el más significativo), y el segundo operando en 5C76h (23670) y 5C77h (23671).

El programa en Assembler puede ser algo como:

```
10   AND  A
20   LD   A,(#5C76)
30   LD   D,A
40   LD   A,(#5CB0)
50   ADC  A,D
60   LD   C,A
70   LD   A,(#5C77)
80   LD   D,A
90   LD   A,(#5CB1)
100  ADC  A,D
110  LD   B,A
120  PUSH AF
130  POP  DE
140  LD   (#5CB0),DE
150  RET
```

La línea 10 pone a «cero» el indicador de acarreo; se trata de un pequeño "truco" que consiste en realizar un "AND" lógico del acumulador consigo mismo, con lo que su contenido no varía, pero no se pone a cero el indicador de acarreo. Más adelante, y dentro de este mismo capítulo, veremos las operaciones lógicas.

Las líneas 20, 30 y 40 cargan los octetos de orden bajo de los dos operandos. La línea 50 los opera (suma) y, si hay acarreo, lo guarda para la suma siguiente. La línea 60 guarda el resultado en "C" (octeto bajo de "BC").

La operación se vuelve a repetir para los octetos altos; las líneas 70, 80 y 90 cargan los operandos. La línea 100 los suma tomando en cuenta el acarreo procedente de la operación anterior. Finalmente, la línea 110 transfiere el resultado al registro "B" (octeto alto de "BC").

más adelante) y le indica si la última operación realizada ha sido una suma o una resta. Por tanto, todas la sumas lo ponen a "0" y todas la restas lo ponen a "1".

"C": Este indicador se activa (valor igual 1) cuando no hay acarreo desde el bit 7, después de la ejecución. Esto ocurre siempre que el valor absoluto del octeto del sustraendo es mayor que el valor absoluto del octeto del minuendo. Es el indicador que se emplea para las instrucciones de restar con acarreo. Interesa observar que este indicador funciona, en la resta, de forma contraria a como lo hace en la suma, es decir, se activa cuando NO hay acarreo en la suma que realiza el microprocesador tras complementar el sustraendo.

"P/V": Este indicador de desbordamiento (overflow) se activa (valor igual 1) siempre que el resultado de la resta haga que el acumulador pase de contener un número menor de 127 a contener uno mayor, o de contener uno mayor de −128 a contener uno menor. Indica, por tanto, un rebosamiento del margen comprendido entre −128 y + 127. Se utiliza como indicador de rebosamiento cuando se trabaja con números en complemento a 2.

-.-.-.-.-.-.

Hata aquí hemos visto las instrucciones que nos han de servir para sumar y restar en código máquina. A continuación veremos las que se encargan de realizar operaciones lógicas tales como AND, OR y XOR. Pero antes, realizaremos unos cuantos ejemplos que podamos ejecutar en el ordenador, y que sirvan para aclarar lo estudiado. También invitamos al lector a que intente resolver los ejercicios que se proponen, y que le darán una medida de cómo va asimilando los conocimientos.

EJEMPLOS:

Al igual que en el capítulo anterior, vamos a hacer algunos programas en código máquina que nos demuestren el funcionamiento de las instrucciones de suma y resta. Al mismo tiempo, iremos cogiendo práctica en la realización y ensamblado de programas en Assembler.

Recomendamos al lector que no se limite a «leer por encima» este curso. Si desea, de verdad, aprender a programar en código máquina, debe seguir el curso encima de una mesa con lápiz y papel en la mano. Intente ensamblar cada programa por usted mismo, y no se limite a ver cómo lo hacemos nosotros; e incluso, atrévase a escribir sus propias rutinas. No se preocupe si el ordenador se le «cuelga» cincuenta veces, es totalmente normal, una rutina en código máquina rara vez funciona a la primera.

Vamos con el primero de nuestros programas. Se trata de sumar dos números sin acarreo. Utilizaremos un programa en Basic que se encargará de gestionar la entrada de datos, llamar a la rutina en C/M e imprimir los resultados, pero la suma la realizaremos en código máquina.

En principio, necesitamos POKEar los dos números que vamos a sumar en dos direcciones de memoria, desde donde serán leídos por la rutina C/M. Estas dos direcciones serán la 5CB0h (23728) para el primer operando, y la 5CB1h (23729) para el segundo; estas direcciones corresponden a una variable del sistema que no se usa.

Primero escribiremos el programa en C/M y luego el Basic. En Assembler, nuestra rutina podría ser algo así:

```
10      LD    A,(#5CB1)
20      LD    B,A
30      LD    A,(#5CB0)
40      ADD   A,B
50      PUSH  AF
60      POP   BC
70      RET
```

Las líneas 10, 20 y 30 leen los dos operandos desde las posiciones de memoria donde los almacenó el Basic. La line 40 realizara la suma equivalente a:

"LET A=A+B"

Las líneas 50 y 60 transfieren el resultado al registro "B" y los indicadores de estado del registro "F", al registro "C". Recuerde que el registro "BC" es lo que nos devuelve USR cuando retornamos a Basic. Mirando las tablas de codificación, podemos esamblar el programa:

Assembler	Hexadecimal	Decimal
LD A,(#5CB1)	: 3A,B1,5C	: 58,177,92
LD B,A	: 47	: 71
LD A,(#5CB0)	: 3A,B0,5C	: 58,176,92
ADD A,B	: 80	: 128
PUSH AF	: F5	: 245
POP BC	: C1	: 193
RET	: C9	: 201

Habría sido interasante que el propio lector hubiera en-

es negativo
pone 0 – en cualquier otro caso
Z ; pone 1 – si el resultado es cero
pone 0 – en cualquier otro caso
H ; pone 1 – si no hay acarreo desde el bit 3
pone 0 – en cualquier otro caso
N ; pone 0 – siempre
C ; pone 1 – si no hay acarreo desde el bit 7
pone 0 – en cualquier otro caso
P/V; pone 1 – si hay desbordamiento (overflow)
pone 0 – en cualquier otro caso

CICLOS DE MEMORIA:

2

CICLOS DE RELOJ:

7

EJEMPLO:

SBC A,40

Valor del registro "A"

(A) 0 0 0 0 0 0 0 0 00h

Indicador del acarreo (C) = 1

Instrucción:

SBC A,40: 1 1 0 1 1 1 1 0 DEh / 0 0 1 0 1 0 0 0 28h

Operación:

```
          n  00101000
         +C         1
            ---------
             00101001

compl. a 2  11010111
      +(A)  00000000
            ---------
            11010111
```

Valor del registro "A" después de la ejecución

(A) 1 1 0 1 0 1 1 1 D7h

Indicadores de condición después de la ejecución.

S	Z		H		P/V	N	C
1	0	x	1	x	1	1	1

Observe que hubo desbordamiento por pasar el registro "A" de un valor positivo a uno negativo.

SBC A,(HL)

OBJETO:

Resta del registro acumulador "A", el valor del octeto de memoria direccionado por el contenido el par de registros HL, más el indicador de acarreo. El resultado se deja en el registro acumulador.

CODIGO DE MAQUINA:

1 0 0 1 1 1 1 0 9Eh

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z ; pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H ; pone 1 - si no hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N ; pone 0 - siempre
C ; pone 1 - si no hay acarreo desde el bit 7
pone 0 en cualquier otro caso
P/V ; pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

2

CICLOS DE RELOJ:

7

EJEMPLO:

SBC A,(HL)

Valor del par de registros "HL"

SBC A,D	9A	154
SBC A,E	9B	155
SBC A,H	9C	156
SBC A,L	9D	157
SBC A,n	DE,n	222,n
SBC A,(HL)	9E	158
SBC A,(IX+d)	DD,9E,d	221,158,d
SBC A,(IY+d)	FD,9E,d	253,158,d

Tabla de codificación para suma y resta.

(H): 11000101 C5h
(L): 00011000 18h

Valor de la posición de memoria C518h

(C518h): 11111111 FFh

Valor del registro "A"

(A): 11111111 FFh

Indicador de acarreo = 0
Instrucción

SBC A,(HL): 10011110 9Eh

Operación:

```
(C518h)  11111111
     +C         0
      ----------
         11111111

compl. a 2  00000001
     +(A)  11111111
      ----------
         00000000
```

Valor del registro "A" después de la ejecución

(A): 00000000 00h

Indicadores de condición después de la ejecución

S	Z		H		P/V	N	C
0	1	x	0	x	1	1	0

Observe que hubo desbordamiento al pasar el registro "A" de un valor negativo a uno positivo.

SBC A,(IX+d)

OBJETO:

Resta al contenido del registro acumulador; el valor del octeto de memoria direccionado por el operando, más el indicador del acarreo. La dirección de memoria se calcula añadiendo al contenido del registro indice IX el valor del entero de desplazamiento "d", el cual puede adquirir los valores desde −128 a +127. El resultado se deja en el registro acumulador.

CODIGO MAQUINA:

11011101	DDh
10011110	9Eh
<—d—>	

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z ; pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H ; pone 1 - si no hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N ; pone 0 - siempre
C ; pone 1 - si no hay acarreo desde el bit 7
pone 0 en cualquier otro caso
P/V ; pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

5

CICLOS DE RELOJ:

19

EJEMPLO:

SBC A,(IX+30)

Valor del registro "IX"

(IX): 11110011 F3h
00011001 19h

Valor de la posición de memoria F337h

(F337h): 01101010 6Ah

Valor del registro "A"

(A): 01111011 7Bh

Indicador de acarreo (C)=1
Instrucción

SBC A,(IX+30): 11011101 DDh
10011110 9Eh
00011110 1Eh

Operación:

```
(F337h)  01101010
     +C         1
      ----------
         01101011

compl. a 2  10010101
     +(A)  01111011
      ----------
         00010000
```

Valor del registro "A" después de la ejecución

(A): 00010000 10h

Indicadores de condición después de la ejecución

S	Z		H		P/V	N	C
0	0	x	0	x	0	1	0

SBC A,(IY+d)

OBJETO:

Resta al contenido del registro acumulador "A", el valor del octeto de la memoria direccionado por el operando, más el indicador de acarreo. La dirección de memoria se calcula añadiendo al contenido del registro indice "IY" el valor del entero de desplazamiento "d", el cual puede adquirir los valores desde −128 a +127. El resultado se deja en el registro acumulador.

CODIGO MAQUINA:

11111101	FDh
10011110	9Eh
<—d—>	

INDICADORES DE CONDICION A LOS QUE AFECTA:

S ; pone 1 - si el resultado es negativo
pone 0 - en cualquier otro caso
Z ; pone 1 - si el resultado es cero
pone 0 - en cualquier otro caso
H ; pone 1 - si no hay acarreo desde el bit 3
pone 0 - en cualquier otro caso
N ; pone 0 - siempre
C ; pone 1 - si no hay acarreo desde el bit 7
pone 0 en cualquier otro caso
P/V ; pone 1 - si hay desbordamiento (overflow)
pone 0 - en cualquier otro caso

CICLOS DE MEMORIA:

5

CICLOS DE RELOJ:

19

EJEMPLO:

SBC A,(IY−1)

Valor del registro "IY"

(IY): 01100110 66h
11100001 E1h

Valor de la posición de memoria 66E0h

(66E0h): 11010011 D3h

Valor del registro "A"

(A): 01100010 62h

Indicador de acarreo (C)=1
Instrucción:

SBC A,(IY−1): 11111101 FDh
10011110 9Eh
11111111 FFh

Operación:

```
(66E0h)  11010011
     +C         1
      ----------
         11010100

Compl. a 2  00101100
     +(A)  01100010
      ----------
         10001110
```

Valor del registro "A" después de la ejecución

(A): 10001110 8Eh

Indicadores de condición después de la ejecución

S	Z		H		P/V	N	C
1	0	x	1	x	1	1	1

Observe que no hubo acarreo desde el bit 7, esto ocurre cuando el sustraendo es menor que el minuendo, como el indicador "C" está invertido en las instrucciones de resta, se pone a "1"; esta situación da como resultado un número negativo. Si se hace el complemento a dos del resultado en el registro "A" nos da el valor 72, que con el signo "−" es el resultado entero de la operación (−72h).

–.–.–.–.–.–.

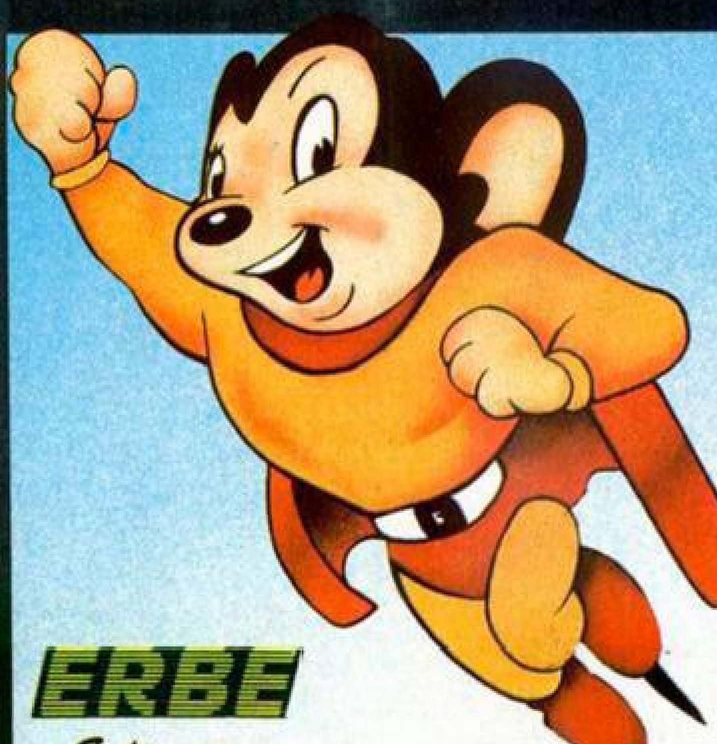
La activación de los indicadores de condición, tanto en la resta (SUB) como en la resta con acarreo (SBC), se hace según las siguientes reglas:

"S": En este indicador se copia el bit 7 del acumulador para indicar si el número que contiene es positivo o negativo.

"Z": Este indicador se activa (valor igual 1) siempre que todos los bit del registro acumulador sean cero después de la ejecución.

"H": Este indicador se activa (valor igual 1) cuando no hay acarreo desde el bit 3 después de la ejecución. Esto ocurre siempre que el valor absoluto de los cuatro bits inferiores del sustraendo es mayor que el valor absoluto de los cuatro inferiores del minuendo.

"N": Este indicador lo toma en cuenta el microprocesador cuando hace un ajuste BCD del acumulador (se verá



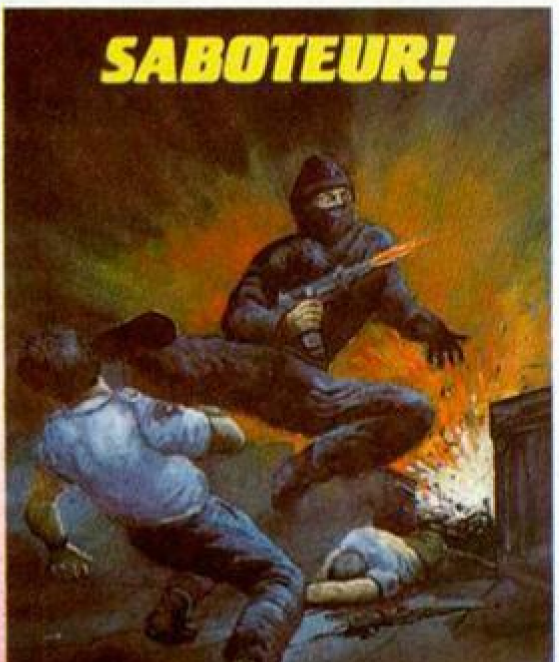
ERBE

Software

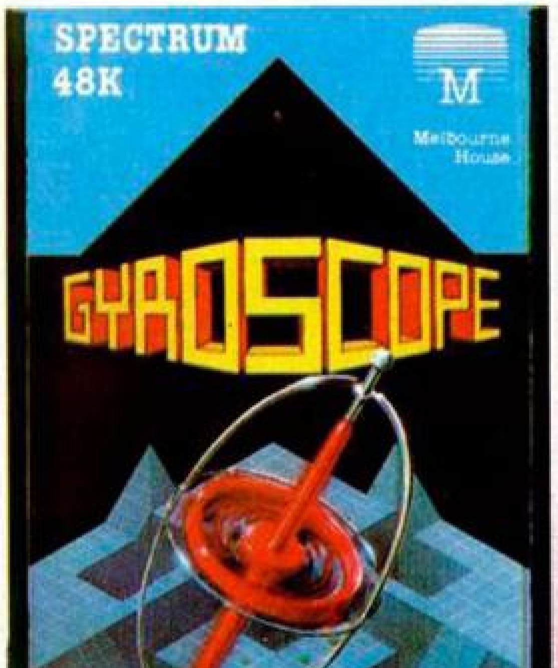
TE INVITA

PREPARA las MALETAS Á Disney World!

COMO LO OYES. EN CADA UNO DE NUESTROS JUEGOS ENCONTRARAS UN CUPON. MANDANOSLO DEBIDAMENTE RELLENADO Y PARTICIPARAS EN EL SORTEO QUE ENTRE TODOS LOS CUPONES RECIBIDOS HAREMOS EL 15 DE ENERO. ¡IMAGINATE! NADA MENOS QUE VIAJE Y ESTANCIA PARA 2 PERSONAS DURANTE 1 SEMANA EN MIAMI, EN DISNEY WORLD.







SABOTEUR

COMO EXPERIMENTADO MERCENARIO CUIDADOSAMENTE ENTRENADO EN ARTES MARCIALES DEBES CUMPLIR LA MISION QUE TE HA SIDO ENCOMENDADA: ROBAR EL DISCO QUE CON LA LISTA DE LOS REBELDES TIENE EL GRAN DICTADOR.

ASTROCLONE

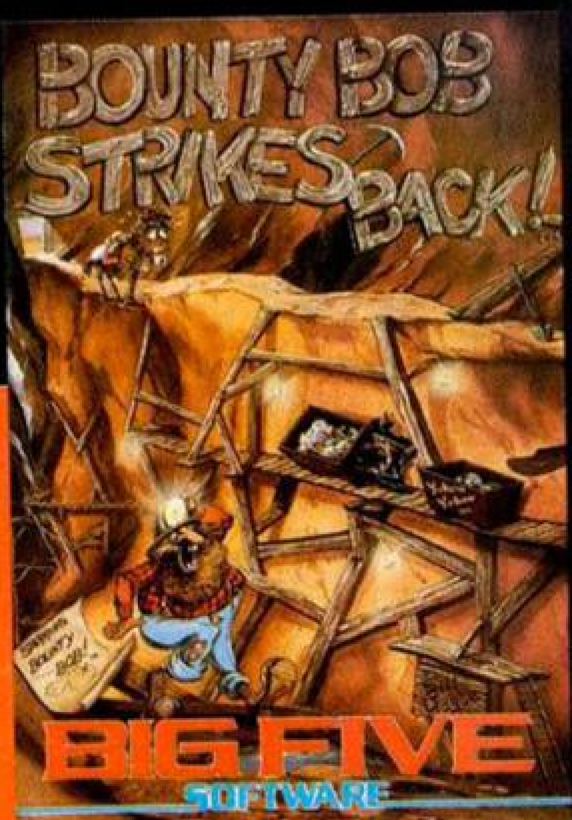
LA PRIMERA AVENTURA INTER-ESTELAR EN 3 DIMENSIONES

GYROSCOPE

¡NO TE LO PIERDAS!. ESTE ES EL CELEBRE "ROLLING" DE LAS MAQUINAS RECREATIVAS. NECESITARAS TODA TU HABILIDAD PARA CONDUCIR LA BOLA A TRAVES DE UN SIN FIN DE PLANOS INCLINADOS Y ESTRECHOS PASILLOS POR UN MONTON DE PANTALLAS.

SI BUSCAS LO MEJOR

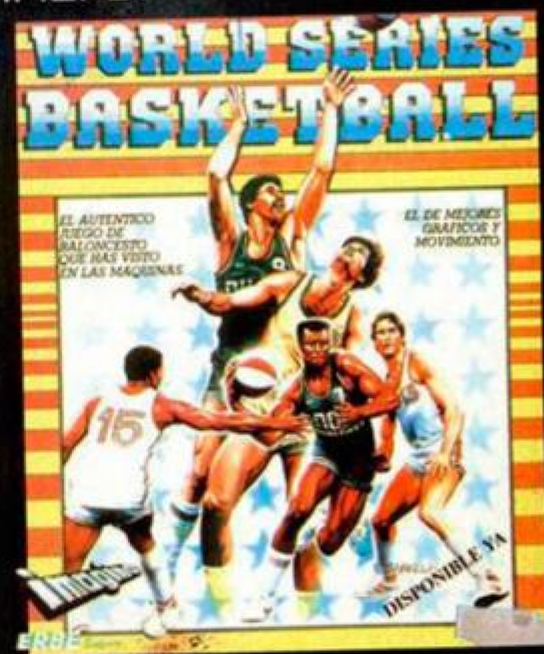
ERBE SOFTWARE. SANTA ENGRACIA, 17



BOUNTY BOB



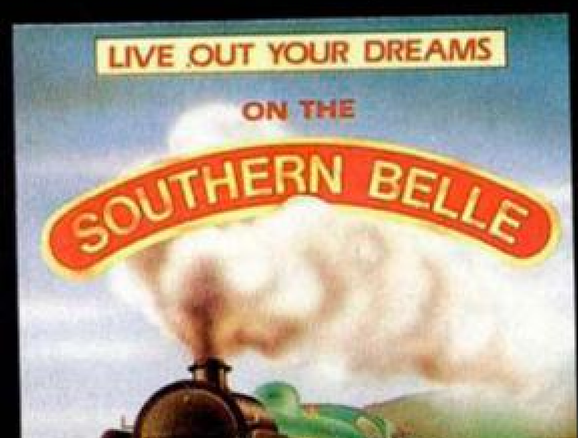
FIGHTING WARRIOR



BASKETBALL



DUMMY RUN



SOUTHERN BELLE



EXPLODN

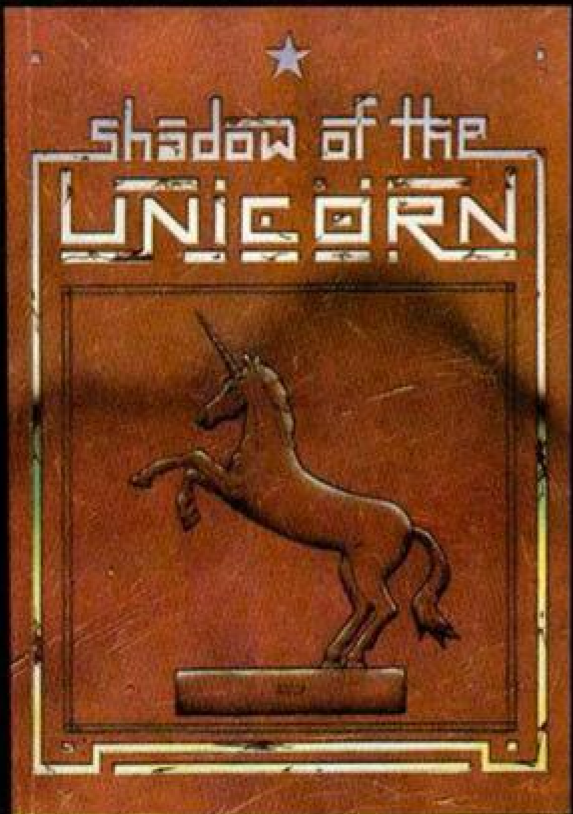
¡REPETIMOS! PARA ENTRAR EN EL SORTIO
DENTRO DE CADA UNO DE ESTOS JJE

Software LO TIENE

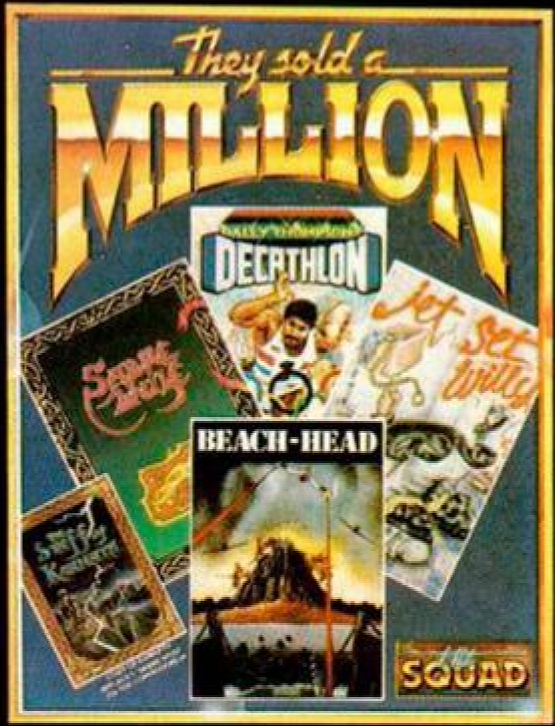
, 17. 28010 MADRID. TFNO.: 447 34 10



CRITICAL MASS



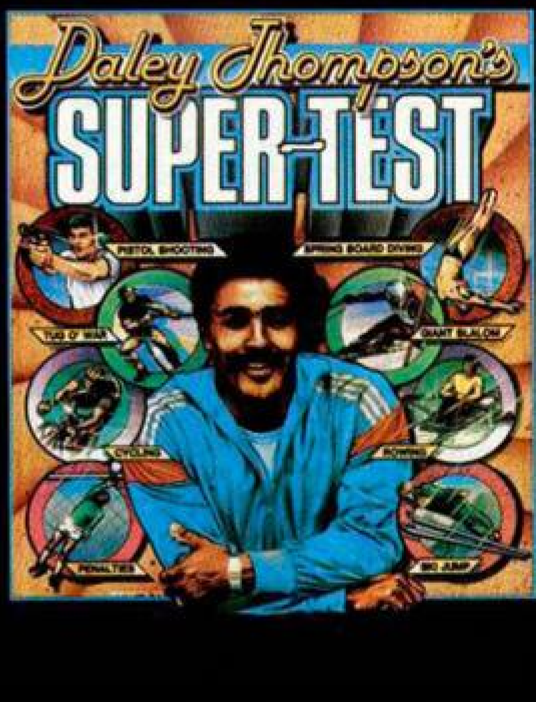
SHADOW OF THE UNICORN



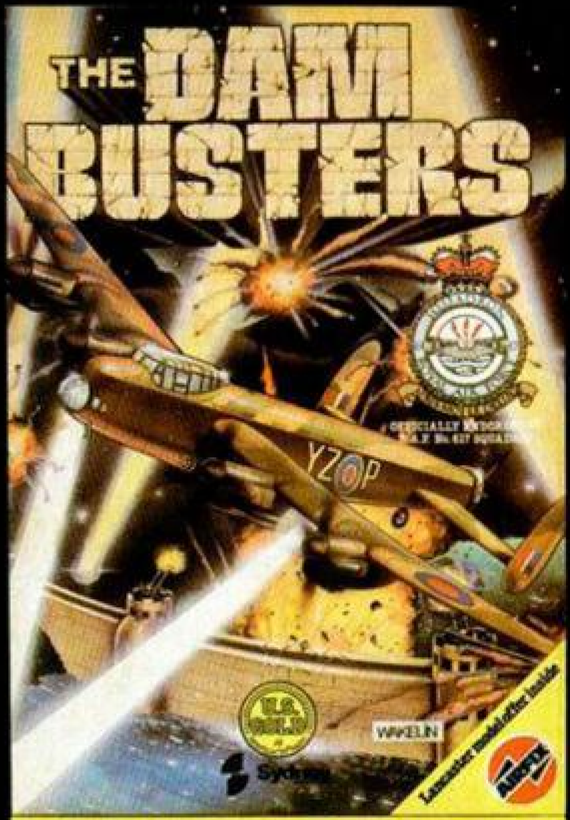
THEY SOLD A MILLION



NG FIST



SUPERTEST



DAMBUSTERS

TEO, RELLENA EL CUPON QUE ENCONTRARAS
JUEGOS Y MANDANOSLO POR CORREO.

Microfile

COMO PASAR A MICRODRIVE LOS PROGRAMAS «TURBO»

Luis E. Juan VIDALES

En la última parte de esta serie os ofrecemos quizás el capítulo más esperado: Conversión de los programas turbo.

h) Los programas «turbo» son reconocibles por su imposibilidad de ser copiados de cassette a cassette con los copiadores tradicionales y su mayor velocidad de grabación de datos, puesta de manifiesto en el tono más agudo del sonido en la cinta y en un tiempo de carga menor que en los programas «normales». En algunos casos, además, las rayas que aparecen en el borde de la pantalla durante la carga son de otro color o inexistentes.

Tal comportamiento es debido a que en la grabación del programa se ha utilizado una rutina de «SAVE» escrita por el propio fabricante en lugar de la contenida en la ROM del SPECTRUM.

Ahora bien, como el programa tiene necesariamente que ser cargado desde un LOAD " " normal ejecutado por el usuario, todo programa «turbo» va precedido de un cargador grabado normalmente, que es el que contiene (casi siempre enmascarada dentro del BASIC) la rutina especial de carga capaz de «digerir» el bloque de código que le sigue.

El proceso que tiene lugar de forma automática, una vez cargado el primer bloque, es el siguiente:

> **PROCESO DE FUNCIONAMIENTO DEL CARGADOR DE UN PROGRAMA TURBO**
>
> – Se descifra y extrae la rutina de carga, inicialmente mezclada con código sin sentido.
> – Se traslada aquélla a una zona libre de memoria (típicamente, el buffer de impresora o el límite superior de la RAM).
> – Se ejecuta la rutina.
> – Una vez cargados los bloques de código, desde ella se salta a la dirección de ejecución.

La clave para la conversión del programa a microdrive consiste en interceptar el salto a la dirección de ejecución, de modo que una vez cargado todo el código en memoria, el cargador nos devuelva el control al BASIC, con lo que dicho código podrá ser almacenado de forma convencional.

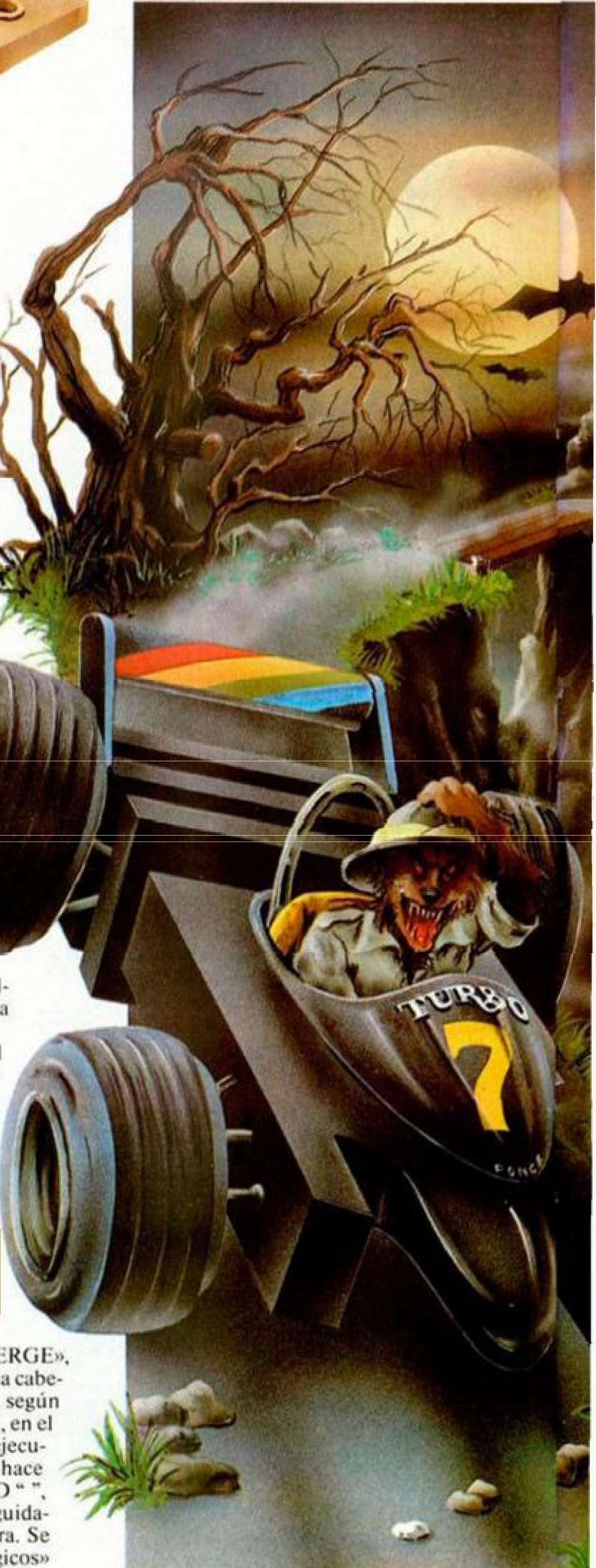
Desgraciadamente, tal interceptación no es en absoluto trivial, puesto que existen casi tantas rutinas de carga especiales como programas «turbo», precisándose un desensamblador, conocimientos de código máquina y buenas dosis de paciencia y suerte para dar con el «POKE mágico». Por otra parte, el propio cargador suele ir protegido de forma que no admite el «MERGE», sus líneas son invisibles y los números de línea son «Ø» y, por si fuera poco, la información contenida en ellas es falsa en ocasiones, estando almacenado el código «útil» en la zona de variables.

Como simple ilustración del método, se expone a continuación el proceso a seguir en la conversión de un conocido programa «turbo»: KNIGHT LORE.

La cabecera del cargador indica lo siguiente:

> Nombre: KNIGHT
> Tipo: BASIC
> Origen: LØ
> Longitud: 1Ø37 bytes

El cargador no admite el «MERGE», por lo que deberemos fabricar una cabecera falsa BASIC de 1Ø37 bytes, según se explicó, en esta misma sección, en el método (a), para cargarlo sin ejecución. Una vez preparada, se hace CLEAR 6ØØØØ : CLS #: LOAD " ", cargando primero la falsa y seguidamente el cargador sin su cabecera. Se detiene la cinta. Los «POKEs mágicos» son:





```
POKE 24849, 195 : POKE 63872, 251 :
POKE 63873, 51 :
POKE 63874, 51 : POKE 63875, 201
```

Con RANDOMIZE USR 24802 la rutina de carga queda descifrada y ubicada limpiamente a partir de la dirección 62000.

Se hace un NEW (la rutina está por encima de RAMTOP) y los POKEs siguientes:

```
POKE 62410, 251 : POKE 62411, 201
```

Gracias a ellos, la rutina devolverá el control al BASIC una vez cargados los dos bloques de código máquina (pantalla de presentación y programa principal).

Se ejecuta la rutina de carga con RANDOMIZE USR 62374 y se pulsa PLAY en el magnetófono. Concluía la carga, el programa se puede almacenar en microdrive con:

```
SAVE * "m";1; "knight" CODE 24832,
30720
```

y el cargador que escribiremos para él será, simplemente:

```
10 LOAD * "m";1; "knight" CODE :
RANDOMIZE USR 24832
```

(Si se quiere guardar también la pantalla de presentación, háganse borde y papel negros y tinta blanca antes de la carga.

Modificación de Cargadores

Los cargadores, según se ha venido citando, son programas en BASIC cuya misión principal es la de cargar las secciones de código máquina que constituyen el verdadero "cuerpo" del programa, así como (generalmente) las pantallas de presentación que entretienen al usuario durante la casi siempre larga carga del programa.

Dada la diferente sintaxis que presenta el almacenamiento en microdrive frente al de cassette, la primera y obligatoria operación a realizar es la sustitución de todos los LOAD " " por LOAD * "m";1; "nombre" y los LOAD " " CODE por el correspondiente LOAD * "m";1; "nombre" CODE origen, longitud (estos dos últimos parámetros no son obligatorios, pero sí recomendables) recordando además que no está permitida la repetición de nombres.

En la mayoría de los casos, no obstante, no merecerá la pena conservar el cargador original modificado, sino que tendremos que reescribirlo para acomodarlo a los eventuales cambios que nos hayamos visto obligados a efectuar en el código (desplazamientos, fraccionamientos, diferentes direcciones de ejecución, etc.). En estos casos es importante el orden en el que se cargan las diferentes secciones, situando el RAMTOP (con un CLEAR XXXXX) por debajo de la sección alta y cargando ésta en primer lugar.

Será frecuente, asimismo, en los programas de las categorías (d) y (e), que el nuevo BASIC no quepa por debajo del RAMTOP debido a los famosos 685 bytes de desplazamiento. En tales casos, convendrá seguir los siguientes procedimientos de ahorro de memoria:

CONSEJOS PARA AHORRAR MEMORIA EN LOS CARGADORES

1) Eliminar todos los REM con comentarios.
2) Eliminar BEEPs y comandos de color innecesarios.
3) Eliminar los orígenes y longitudes innecesarios de los LOAD...CODE
4) Sustituir el número 0 por NOT PI y el 1 por PI/PI
5) Sustituir el resto de los números por el valor de la cadena de caracteres que los representa (ejemplo: cambiar 24580 por VAL "24580")
6) "In extremis", eliminar el CLEAR XXXXX que fija el RAMTOP conflictivo. Recuérdese que en este caso se corre el riesgo de "corromper" el stack del Z80, perdiendo el control. Conviene, pues, hacer la prueba con el código y el cartucho de microdrive puestos en lugar seguro.

En la tabla adjunta se indica la estructura de los cargadores desde Microdrive, según sea la categoría del programa o Código Máquina, como explicábamos en semanas anteriores:

ESTRUCTURA DE LOS CARGADORES DESDE MICRODRIVE

Tipo de programa	
a, b)	10 LOAD * "m";1;"nombre"
c)	10 LOAD * "m";1;"nombre" SCREEN$
d)	10 CLEAR origen − 1
	20 LOAD * "m";1;"nombre" CODE origen, longitud
	30 RANDOMIZE USR dirección de ejecución
e)	10 CLEAR origen actual − 1
	20 LOAD * "m";1;"nombre", CODE origen actual, longitud
	30 RANDOMIZE USR nueva dirección de ejecución
f)	10 CLEAR origen alto − 1
	20 LOAD * "m";1;"secc alta" CODE orig. alto, long. a
	30 LOAD * "m";1;"secc baja" CODE orig. bajo, long. b
	40 RANDOMIZE USR nueva dirección de ejecución
g, h)	Una vez transformados, les corresponderá un cargador tipo (d) a (f).

EL MICROPROCESADOR Z-80 (y V)

Primitivo de FRANCISCO

Concluimos con esta quinta parte la serie que hemos dedicado al conocimiento del microprocesador Z-80. Con estos artículos hemos pretendido de modo divulgativo dar a conocer como funciona este microprocesador siempre desde la perspectiva hardware con vistas hacia la conexión de periféricos especialmente.

El SPECTRUM viene equipado con memorias RAM dinámicas, gracias a las prestaciones que tiene el Z-80 para este cometido. Si se hubieran incluido estáticas su consumo sería del orden de cinco veces superior junto con una alta necesidad de disipación de calor y un gran espacio en la tarjeta.

Para conectar RAM dinámicas al Z-80 sólo hay que adosarle unos cuantos componentes. En la figura 3 de la semana pasada se incluía un dibujo en el que de manera sinoptica se veía la filosofía de conexionado. Existe un bloque multiplexor encargado de doblar el bus de direcciones bajo la acción de la señal MREQ que en el ciclo de refresco actúa de clock, al tiempo que, con un cierto retardo activa las columnas de refresco en la organización interna de la memoria.

Cuando la señal RFSH se va a nivel bajo, el contenido del registro interno I pasa los bits bajos del bus de direcciones para direccionar las columnas de refresco. Este registro se incrementa en uno constantemente en cada ciclo de refresco pasando de FF a ØØ nuevamente como un estado más.

Las células de retardo incluidas en la línea MREQ sirven para evitar solapamientos de los flancos en las señales de refresco. En el ejemplo de la figura el BIT A12 se usa para direccionar una u otra RAM de 4 Kbytes (obsérvese que la señal RAS en el direccionamiento de acceso sólo es permitida cuando no existe RFSH y se produce MREQ. La señal RAS habilita la dirección a las filas según la organización interna en filas y columnas.

Las instrucciones del Z-80

El Z-80 posee 256 instrucciones que se decodifican únicamente por el primer byte de código. Así, por ejemplo, la instrucción "DJNZ,e" es de dos bytes 1Ø xx de los cuales 1Ø es el byte de código y xx es el desplazamiento en el salto relativo que representa.

En la figura número 1 se muestra la tabla con todas las instrucciones de este tipo en sus diferentes variaciones según el registro que interviene en el proceso de cada una de ellas.

La tabla está en forma de doble entrada de manera que el byte de código en hexadecimal se forma mediante la numeración de filas y columnas.

La finalidad de esta tabla es la de mostrar de forma esquemática la estructuración de las instrucciones del Z-80 mediante la cual se rige el decodificador de instrucciones entorno al microprocesador. En la tabla se puede observar la ley de formación de las instrucciones y la secuencia que se siguen con respecto a los registros.

En cada cuadrado de la tabla se muestra: El código de la instrucción que se ha subrayado, y el nemónico o nombre por el que se le conoce y el que ha de usarse en los programas fuente. Sobre el nombre y función de las instrucciones no trataremos porque desde hace algunas semanas se está incluyendo en Microhobby un curso de programación en código máquina en donde se explican todas ellas profundamente. Esta tabla puede ser un útil como complemento para aquéllos que hayan decidido seguir el citado curso de las páginas centrales.

En algunas casillas existen unas letras minúsculas que significan:

e = Byte de desplazamiento
dd = Byte de dato
dddd = Doble byte de dato
addr = Dirección absoluta

Esta tabla fue la que confeccionó el fabricante del Z-80 a la hora de distribuir las instrucciones a los 256 códigos posibles con 8 bits. Pero su intención era la de dar más potencia a este microprocesador, por lo que hubo de ampliar a dos bytes el código de las instrucciones extras. Para ello eligió cuatro byts de código (CB, DD, ED y FD) que los empleó como ventanas que comunican con otras tantas expansiones.

En la tabla 1 mostramos las instrucciones que se corresponden con el byte CB por el cual empiezan. Todas ellas son de dos bytes, el segundo byte cubre casi otras tantas

	Ø	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Ø	00 NOP	01 LD BC,+dddd	02 LD (BC),A	03 INC BC	04 INC B	05 DEC B	06 LD B,+dd	07 RLCA	08 EX AF,A'F'	09 ADD HL,BC	0A LD A,(BC)	0B DEC BC	0C INC C	0D DEC C	0E LD C,+dd	0F RRCA
1	10 DJNZ,e	11 LD DE,+dddd	12 LD (DE),A	13 INC DE	14 INC D	15 DEC D	16 LD D,+dd	17 RLA	18 JR,e	19 ADD HL,DE	1A LD A,(DE)	1B DEC DE	1C INC E	1D DEC E	1E LD E,+dd	1F RRA
2	20 JR NZ,e	21 LD HL,+dddd	22 LD (addr),HL	23 INC HL	24 INC H	25 DEC H	26 LD H,+dd	27 DAA	28 JR Z,e	29 ADD HL,HL	2A LD HL,(addr)	2B DEC HL	2C INC L	2D DEC L	2E LD L,+dd	2F CPL
3	30 JR NC,e	31 LD SP,+dddd	32 LD (addr),A	33 INC SP	34 INC (HL)	35 DEC (HL)	36 LD (HL),dd	37 SCF	38 JR C,e	39 ADD HL,SP	3A LD A,(addr)	3B DEC SP	3C INC A	3D DEC A	3E LD A,+dd	3F CCF
4	40 LD B,B	41 LD B,C	42 LD B,D	43 LD B,E	44 LD B,H	45 LD B,L	46 LD B,(HL)	47 LD B,A	48 LD C,B	49 LD C,C	4A LD C,D	4B LD C,E	4C LD C,H	4D LD C,L	4E LD C,(HL)	4F LD C,A
5	50 LD D,B	51 LD D,C	52 LD D,D	53 LD D,E	54 LD D,H	55 LD D,L	56 LD D,(HL)	57 LD D,A	58 LD E,B	59 LD E,C	5A LD E,D	5B LD E,E	5C LD E,H	5D LD E,L	5E LD E,(HL)	5F LD E,A
6	60 LD H,B	61 LD H,C	62 LD H,D	63 LD H,E	64 LD H,H	65 LD H,L	66 LD H,(HL)	67 LD H,A	68 LD L,B	69 LD L,C	6A LD L,D	6B LD L,E	6C LD L,H	6D LD L,L	6E LD L,(HL)	6F LD L,A
7	70 LD (HL),B	71 LD (HL),C	72 LD (HL),D	73 LD (HL),E	74 LD (HL),H	75 LD (HL),L	76 HALT	77 LD (HL),A	78 LD A,B	79 LD A,C	7A LD A,D	7B LD A,E	7C LD A,H	7D LD A,L	7E LD A,(HL)	7F LD A,A
8	80 ADD A,B	81 ADD A,C	82 ADD A,D	83 ADD A,E	84 ADD A,H	85 ADD A,L	86 ADD A,(HL)	87 ADD A,A	88 ADC A,B	89 ADC A,C	8A ADC A,D	8B ADC A,E	8C ADC A,H	8D ADC A,L	8E ADC A,(HL)	8F ADC A,A
9	90 SUB B	91 SUB C	92 SUB D	93 SUB E	94 SUB H	95 SUB L	96 SUB (HL)	97 SUB A	98 SBC A,B	99 SBC A,C	9A SBC A,D	9B SBC A,E	9C SBC A,H	9D SBC A,L	9E SBC A,(HL)	9F SBC A,A
A	A0 AND B	A1 AND C	A2 AND D	A3 AND E	A4 AND H	A5 AND L	A6 AND (HL)	A7 AND A	A8 XOR B	A9 XOR C	AA XOR D	AB XOR E	AC XOR H	AD XOR L	AE XOR (HL)	AF XOR A
B	B0 OR B	B1 OR C	B2 OR D	B3 OR E	B4 OR H	B5 OR L	B6 OR (HL)	B7 OR A	B8 CP B	B9 CP C	BA CP D	BB CP E	BC CP H	BD CP L	BE CP (HL)	BF CP A
C	C0 RET NZ	C1 POP BC	C2 JP NZ,addr	C3 JP addr	C4 CALL NZ,addr	C5 PUSH BC	C6 ADD A,+dd	C7 RST 0000	C8 RET Z	C9 RET	CA JP Z,addr	CB TABLA 1	CC CALL Z,addr	CD CALL addr	CE ADC A,+dd	CF RST 0008
D	D0 RET NC	D1 POP DE	D2 JP NC,addr	D3 OUT (+dd),A	D4 CALL NC,addr	D5 PUSH DE	D6 SUB +dd	D7 RST 0010	D8 RET C	D9 EXX	DA JP C,addr	DB IN A,(+dd)	DC CALL C,addr	DD TABLA 2	DE SBC A,+dd	DF RST 0018
E	E0 RET PO	E1 POP HL	E2 JP PO,addr	E3 EX (SP),HL	E4 CALL PO,addr	E5 PUSH HL	E6 AND +dd	E7 RST 0020	E8 RET PE	E9 JP (HL)	EA JP PE,addr	EB EX DE,HL	EC CALL PE,addr	ED TABLA 3	EE XOR +dd	EF RST 0028
F	F0 RET P	F1 POP AF	F2 JP P,addr	F3 DI	F4 CALL P,addr	F5 PUSH AF	F6 OR +dd	F7 RST 0030	F8 RET M	F9 LD SP,HL	FA JP M,addr	FB EI	FC CALL M,addr	FD TABLA 4	FE CP +dd	FF RST 0038

Figura 1. En esta tabla se muestran los códigos de las instrucciones desde ØØ hast FF. Los recuadros marcados corresponden a las Tablas 1, 2, 3 y 4 en que se detallan las instrucciones de expansión.

INSTRUCCIONES CB

CB00	RLC B	CB5A	BIT 3, D	CBAD	RES 5, L
CB01	RLC C	CB5B	BIT 3, E	CBAE	RES 5, (HL)
CB02	RLC D	CB5C	BIT 3, H	CBAF	RES 5, A
CB03	RLC E	CB5D	BIT 3, L	CBB0	RES 6, B
CB04	RLC H	CB5E	BIT 3, (HL)	CBB1	RES 6, C
CB05	RLC L	CB5F	BIT 3, A	CBB2	RES 6, D
CB06	RLC (HL)	CB60	BIT 4, B	CBB3	RES 6, E
CB07	RLC A	CB61	BIT 4, C	CBB4	RES 6, H
CB08	RRC B	CB62	BIT 4, D	CBB5	RES 6, L
CB09	RRC C	CB63	BIT 4, E	CBB6	RES 6, (HL)
CB0A	RRC D	CB64	BIT 4, H	CBB7	RES 6, A
CB0B	RRC E	CB65	BIT 4, L	CBB8	RES 7, B
CB0C	RRC H	CB66	BIT 4, (HL)	CBB9	RES 7, C
CB0D	RRC L	CB67	BIT 4, A	CBBA	RES 7, D
CB0E	RRC (HL)	CB68	BIT 5, B	CBBB	RES 7, E
CB0F	RRC A	CB69	BIT 5, C	CBBC	RES 7, H
CB10	RL B	CB6A	BIT 5, D	CBBD	RES 7, L
CB11	RL C	CB6B	BIT 5, E	CBBE	RES 7, (HL)
CB12	RL D	CB6C	BIT 5, H	CBBF	RES 7, A
CB13	RL E	CB6D	BIT 5, L	CBC0	SET 0, B
CB14	RL H	CB6E	BIT 5, (HL)	CBC1	SET 0, C
CB15	RL L	CB6F	BIT 5, A	CBC2	SET 0, D
CB16	RL (HL)	CB70	BIT 6, B	CBC3	SET 0, E
CB17	RL A	CB71	BIT 6, C	CBC4	SET 0, H
CB18	RR B	CB72	BIT 6, D	CBC5	SET 0, L
CB19	RR C	CB73	BIT 6, E	CBC6	SET 0, (HL)
CB1A	RR D	CB74	BIT 6, H	CBC7	SET 0, A
CB1B	RR E	CB75	BIT 6, L	CBC8	SET 1, B
CB1C	RR H	CB76	BIT 6, (HL)	CBC9	SET 1, C
CB1D	RR L	CB77	BIT 6, A	CBCA	SET 1, D
CB1E	RR (HL)	CB78	BIT 7, B	CBCB	SET 1, E
CB1F	RR A	CB79	BIT 7, C	CBCC	SET 1, H
CB20	SLA B	CB7A	BIT 7, D	CBCD	SET 1, L
CB21	SLA C	CB7B	BIT 7, E	CBCE	SET 1, (HL)
CB22	SLA D	CB7C	BIT 7, H	CBCF	SET 1, A
CB23	SLA E	CB7D	BIT 7, L	CBD0	SET 2, B
CB24	SLA H	CB7E	BIT 7, (HL)	CBD1	SET 2, C
CB25	SLA L	CB7F	BIT 7, A	CBD2	SET 2, D
CB26	SLA (HL)	CB80	RES 0, B	CBD3	SET 2, E
CB27	SLA A	CB81	RES 0, C	CBD4	SET 2, H
CB28	SRA B	CB82	RES 0, D	CBD5	SET 2, L
CB29	SRA C	CB83	RES 0, E	CBD6	SET 2, (HL)
CB2A	SRA D	CB84	RES 0, H	CBD7	SET 2, A
CB2B	SRA E	CB85	RES 0, L	CBD8	SET 3, B
CB2C	SRA H	CB86	RES 0, (HL)	CBD9	SET 3, C
CB2D	SRA L	CB87	RES 0, A	CBDA	SET 3, D
CB2E	SRA (HL)	CB88	RES 1, B	CBDB	SET 3, E
CB2F	SRA A	CB89	RES 1, C	CBDC	SET 3, H
CB38	SRL B	CB8A	RES 1, D	CBDD	SET 3, L
CB39	SRL C	CB8B	RES 1, E	CBDE	SET 3, (HL)
CB3A	SRL D	CB8C	RES 1, H	CBDF	SET 3, A
CB3B	SRL E	CB8D	RES 1, L	CBE0	SET 4, B
CB3C	SRL H	CB8E	RES 1, (HL)	CBE1	SET 4, C
CB3D	SRL L	CB8F	RES 1, A	CBE2	SET 4, D
CB3E	SRL (HL)	CB90	RES 2, B	CBE3	SET 4, E
CB3F	SRL A	CB91	RES 2, C	CBE4	SET 4, H
CB40	BIT 0, B	CB92	RES 2, D	CBE5	SET 4, L
CB41	BIT 0, C	CB93	RES 2, E	CBE6	SET 4, (HL)
CB42	BIT 0, D	CB94	RES 2, H	CBE7	SET 4, A
CB43	BIT 0, E	CB95	RES 2, L	CBE8	SET 5, B
CB44	BIT Ø, H	CB96	RES 2, (HL)	CBE9	SET 5, C
CB45	BIT 0, L	CB97	RES 2, A	CBEA	SET 5, D
CB46	BIT 0, (HL)	CB98	RES 3, B	CBEB	SET 5, E
CB47	BIT 0, A	CB99	RES 3, C	CBEC	SET 5, H
CB48	BIT 1, B	CB9A	RES 3, D	CBED	SET 5, L
CB49	BIT 1, C	CB9B	RES 3, E	CBEE	SET 5, (HL)
CB4A	BIT 1, D	CB9C	RES 3, H	CBEF	SET 5, A
CB4B	BIT 1, E	CB9D	RES 3, L	CBF0	SET 6, B
CB4C	BIT 1, H	CB9E	RES 3, (HL)	CBF1	SET 6, C
CB4D	BIT 1, L	CB9F	RES 3, A	CBF2	SET 6, D
CB4E	BIT 1, (HL)	CBA0	RES 4, B	CBF3	SET 6, E
CB4F	BIT 1, A	CBA1	RES 4, C	CBF4	SET 6, H
CB50	BIT 2, B	CBA2	RES 4, D	CBF5	SET 6, L
CB51	BIT 2, C	CBA3	RES 4, E	CBF6	SET 6, (HL)
CB52	BIT 2, D	CBA4	RES 4, H	CBF7	SET 6, A
CB53	BIT 2, E	CBA5	RES 4, L	CBF8	SET 7, B
CB54	BIT 2, H	CBA6	RES 4, (HL)	CBF9	SET 7, C
CB55	BIT 2, L	CBA7	RES 4, A	CBFA	SET 7, D
CB56	BIT 2, (HL)	CBA8	RES 5, B	CBFB	SET 7, E
CB57	BIT 2, A	CBA9	RES 5, C	CBFC	SET 7, H
CB58	BIT 3, B	CBAA	RES 5, D	CBFD	SET 7, L
CB59	BIT 3, C	CBAB	RES 5, E	CBFE	SET 7, (HL)
		CBAC	RES 5, H	CBFF	SET 7, A

Tabla 1. Expansión de las instrucciones de 2 bytes que comienzan por CB.

INSTRUCCIONES DD

DD09	ADD IX, BC
DD19	ADD IX, DE
DD21XXXX	LD IX, NN
DD22XXXX	LD (NN), IX
DD23	INC IX
DD29	ADD IX, IX
DD2AXXXX	LD IX, (NN)
DD2B	DEC IX
DD34XX	INC (IX + d)
DD35XX	DEC (IX + d)
DD36XXXX	LD (IX + d), N
DD39	ADD IX, SP
DD46XX	LD B, (IX + d)
DD4EXX	LD C, (IX + d)
DD56XX	LD D, (IX + d)
DD5EXX	LD E, (IX + d)
DD66XX	LD H, (IX + d)
DD6EXX	LD L, (IX + d)
DD70XX	LD (IX + d), B
DD71XX	LD (IX + d), C
DD72XX	LD (IX + d), D
DD73XX	LD (IX + d), E
DD74XX	LD (IX + d), H
DD75XX	LD (IX + d), L
DD77XX	LD (IX + d), A
DD7EXX	LD A, (IX + d)
DD86XX	ADD A, (IX + d)
DD8EXX	ADC A, (IX + d)
DD9EXX	SUB (IX + d)
DD9EXX	SBC A, (IX + d)
DDA6XX	AND (IX + d)
DDAEXX	XOR (X + d)
DDB6XX	OR (IX + d)
DDBEXX	CP (IX + d)
DDE1	POP IX
DDE3	EX (SP), IX
DDE5	PUSH IX
DDE9	JP (IX)
DDF9	LD SP, IX
DDCBXX06	RLC (IX + d)
DDCBXX0E	RRC (IX + d)
DDCBXX16	RL (IX + d)
DDCBXX1E	RR (IX + d)
DDCBXX26	SLA (IX + d)
DDCBXX2E	SRA (IX + d)
DDCBXX3E	SRL (IX + d)
DDCBXX46	BIT 0, (IX + d)
DDCBXX4E	BIT 1, (IX + d)
DDCBXX56	BIT 2, (IX + d)
DDCBXX5E	BIT 3, (IX + d)
DDCBXX66	BIT 4, (IX + d)
DDCBXX6E	BIT 5, (IX + d)
DDCBXX76	BIT 6, (IX + d)
DDCBXX7E	BIT 7, (IX + d)
DDCBXX86	RES 0, (IX + d)
DDCBXX8E	RES 1, (IX + d)
DDCBXX96	RES 2, (IX + d)
DDCBXX9E	RES 3, (IX + d)
DDCBXXA6	RES 4, (IX + d)
DDCBXXAE	RES 5, (IX + d)
DDCBXXB6	RES 6, (IX + d)
DDCBXXBE	RES 7, (IX + d)
DDCBXXC6	SET 0, (IX + d)
DDCBXXCE	SET 1, (IX + d)
DDCBXXD6	SET 2, (IX + d)
DDCBXXDE	SET 3, (IX + d)
DDCBXXE6	SET 4, (IX + d)
DDCBXXEE	SET 5, (IX + d)
DDCBXXF6	SET 6, (IX + d)
DDCBXXFE	SET 7, (IX + d)

Tabla 2. Expansión de las instrucciones de 2, 3 y 4 bytes que comienzan por DD.

INSTRUCCIONES FD

FD09	ADD IY, BC
FD19	ADD IY, DE
FD21XXXX	LD IY, NN
FD22XXXX	LD (NN), IY
FD23	INC IY
FD29	ADD IY, IY
FD2AXXXX	LD IY, (NN)
FD2B	DEC IY
FD34XX	INC (IY + d)
FD35XX	DEC (IY + d)
FD36XXXX	LD (IY + d), N
FD39	ADD IY, SP
FD46XX	LD B, (IY + d)
FD4EXX	LD C, (IY + d)
FD56XX	LD D, (IY + d)
FD5EXX	LD E, (IY + d)
FD66XX	LD H, (IY + d)
FD6EXX	LD L, (IY + d)
FD70XX	LD (IY + d), B
FD71XX	LD (IY + d), C
FD72XX	LD (IY + d), D
FD73XX	LD (IY + d), E
FD74XX	LD (IY + d), H
FD75XX	LD (IY + d), L
FD77XX	LD (IY + d), A
FD7EXX	LD A, (IY + d)
FD86XX	ADD A, (IY + d)
FD8EXX	ADC A, (IY + d)
FD96XX	SUB (IY + d)
FD9EXX	SBC A, (IY + d)
FDAEXX	AND (IY + d)
FDAEXX	XOR (IY + d)
FDB6XX	OR (IY + d)
FDBEXX	CP (IY + d)
FDE1	POP IY
FDE3	EX (SP), IY
FDE5	PUSH IY
FDE9	JP (IY)
FDF9	LD SP, IY
FDCBXX06	RLC (IY + d)
FDCBXX0E	RRC (IY + d)
FDCBXX16	RL (IY + d)
FDCBXX1E	RR (IY + d)
FDCBXX26	SLA (IY + d)
FDCBXX2E	SRA (IY + d)
FDCBXX3E	SRL (IY + d)
FDCBXX46	BIT 0, (IY + d)
FDCBXX4E	BIT 1, (IY + d)
FDCBXX56	BIT 2, (IY + d)
FDCBXX5E	BIT 3, (IY + d)
FDCBXX66	BIT 4, (IY + d)
FDCBXX6E	BIT 5, (IY + d)
FDCBXX76	BIT 6, (IY + d)
FDCBXX7E	BIT 7, (IY + d)
FDCBXX86	RES 0, (IY + d)
FDCBXX8E	RES 1, (IY + d)
FDCBXX96	RES 2, (IY + d)
FDCBXX9E	RES 3, (IY + d)
FDCBXXA6	RES 4, (IY + d)
FDCBXXAE	RES 5, (IY + d)
FDCBXXB6	RES 6, (IY + d)
FDCBXXBE	RES 7, (IY + d)
FDCBXXC6	SET 0, (IY + d)
FDCBXXCE	SET 1, (IY + d)
FDCBXXD6	SET 2, (IY + d)
FDCBXXDE	SET 3, (IY + d)
FDCBXXE6	SET 4, (IY + d)
FDCBXXEE	SET 5, (IY + d)
FDCBXXF6	SET 6, (IY + d)
FDCBXXFE	SET 7, (IY + d)

Tabla 4. Expansión de las instrucciones de 2, 3 y 4 bytes que comienzan por FD.

256 posibilidades (desde ØØ hasta FF), aunque entre 2F y 38 se observa una discontinuidad de 8 bytes, así pues las instrucciones CB 3Ø hasta CB 37 no existen.

En las tablas 2, 3 y 4 se muestran la expansión de las instrucciones que empiezan por DD, ED y FD. Algunas de éstas pueden llegar hasta 4 bytes, éstas son las que tratan los registros IX e IY.

En las tablas el signo X simboliza, un dígito de un dato o un desplazamiento.

Algunas de las instrucciones que tratan los registros IX e IY en las tablas 2 y 4 requieren de hasta tres bytes de código. Esto representa una rara particularidad del decodificador de instrucción que además incluye el byte de desplazamiento entre el segundo y tercer byte de código.

Obsérvese que las expansiones DD, ED y FD no usan todas las combinaciones del byte que le sigue, sino algunas que le han sido propicias al diseñador por ejemplo la instrucción DD 24 no existe.

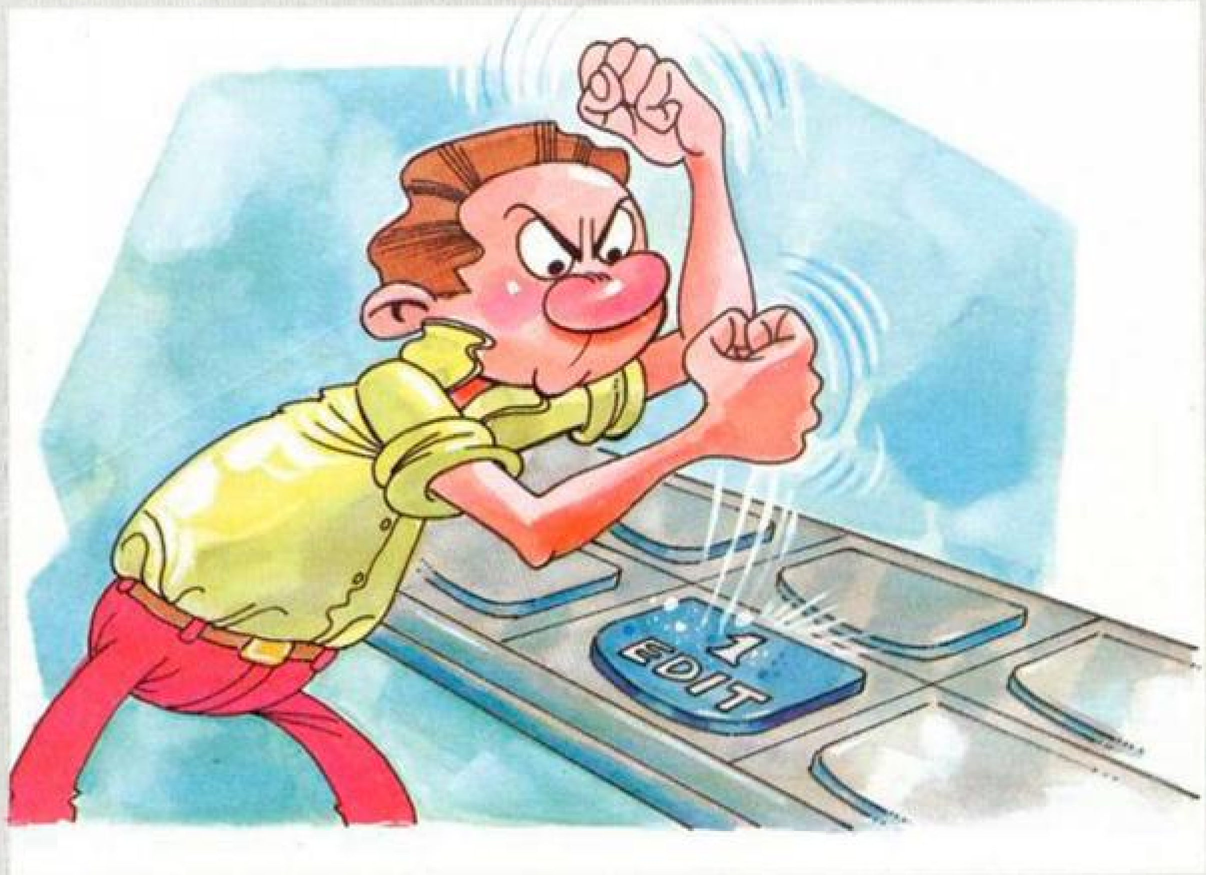
Este sistema de ampliación de instrucciones es original del Z-80 y es uno de los factores que le proporcionan su gran potencia, aunque hay que reconocer que esto es a costa de una mayor lentitud operativa al tratar estas instrucciones, pues ha de procesar hasta cuatro bytes de código.

INSTRUCCIONES ED

ED57	LD A, I	ED44	NEG	ED67	RRD	EDA2	INI
ED58	IN E, (C)	ED45	RETN	ED68	IN L, (C)	EDA3	OUTI
ED59	OUT (C), E	ED46	IM 0	ED69	OUT (C), L	EDA8	LDD
ED5A	ADC HL, DE	ED47	LD I, A	ED6A	ADC HL, HL	EDA9	CPD
ED5BXXXX	LD DE, (NN)	ED48	IN C, (C)	ED6F	RLD	EDAA	IND
ED5E	IM 2	ED49	OUT (C), C	ED72	SBC HL, SP	EDAB	OUTD
ED60	IN H, (C)	ED4A	ADC HL, BC	ED73XXXX	LD (NN), SP	EDB0	LDIR
ED61	OUT (C), H	ED4BXXXX	LD BC, (NN)	ED78	IN A, (C)	EDB1	CPIR
ED62	SBC HL, HL	ED4D	RETI	ED79	OUT (C), A	EDB2	INIR
ED40	IN B, (C)	ED50	IN D, (C)	ED7A	ADC HL, SP	EDB3	OTIR
ED41	OUT (C), B	ED51	OUT (C), D	ED7BXXXX	LD SP, (NN)	EDB8	LDDR
ED42	SBC HL, BC	ED52	SBC HL, DE	EDA0	LDI	EDB9	CPDR
ED43XXXX	LD (NN), BC	ED53XXXX	LD (NN), DE	EDA1	CPI	EDBA	INDR
		ED56	IM 1			EDBB	OTDR

Tabla 3. Expansión de las instrucciones de 2 y 4 bytes que comienzan por ED.

TRUCOS



ANULAR LA FUNCION EDIT

Mediante la sentencia

POKE 23572,32

que nos manda Angel Abad, podremos anular la tecla EDIT, con lo cual evitaremos que sean editadas las líneas de los programas.

FUNCION MID$

Todo el mundo sabe que la función MID$ (<variable de cadena> . <exp. núm. — 2>) extrae de la variable de cadena tantos caracteres a partir de expresión numérica —1 como se nos indique en expresión numérica —2. La simulación de esta sentencia con DEF FN, que nos manda Rafael Remartinez es de la siguiente manera:

```
   5 INPUT "dar la posicion ";p
   6 INPUT "dar el numero de car
acteres  ";n
   8 INPUT "dar la variable de c
adena  ";x$
  10 DEF FN m$(x$,p,n)=x$(p TO p
+n-1)
  20 PRINT FN m$(x$,p,n)
```

ELIMINAR LA LINEA 2

Para eliminar la línea 2 sólo hay que introducir esta pequeña subrutina de Alberto Casals en el Spectrum y guardarla mediante un SAVE «nombre» LINE 2.

```
   2 POKE 23635,PEEK 23637: POKE
23363,PEEK 23638
  10 REM TOP SECRET!
```

EFECTOS RAYADOS

Como hay efectos para todos los gustos, aquí os

```
  10 BORDER 6: PAPER 5: INK 1: C
LS
  20 FOR F=0 TO RND*175
  30 PLOT 0,F
  40 DRAW RND*255,0
  50 NEXT F
  60 GO TO 20
```

ofrecemos uno de Carlos Neira que, seguro, os gustará.

Además nos incluye otras posibilidades para hacer un BREAK, apretando simultáneamente CAPS SHIFT, SYMBOL SHIFT, X, C, V, B o CAPS SHIFT, SYMBOL SHIFT, comillas, punto y coma.

PASAR DE BINARIO A DECIMAL

La utilidad de este truco que nos manda Rafael Lobo consiste en que al introducir números en base dos, sin necesidad de utilizar la sentencia BIN, el ordenador nos los trasbasa a base diez convirtiéndolos en decimal. El programa puede ser modificado para obtener otros resultados, a gusto del consumidor.

```
  10 REM BINARIO-DECIMAL
  20 CLS
  30 PRINT "Binario","Decimal"
  40 PRINT
  50 INPUT "Binario:"; LINE A$
  60 LET T=0
  70 LET K=LEN A$
  80 FOR N=1 TO K
  90 LET B$=A$(N)
 100 LET A=VAL B$*2↑(K-N)
 110 LET T=T+A
 120 PRINT B$;
 130 NEXT N
 140 PRINT TAB 16;T
 150 GO TO 50
```

LOS JUSTICIEROS DEL SOFTWARE

FIGHTING WARRIOR

Bueno, aquí está ya nuestro primer juicio. En esta ocasión le ha tocado la china al «Fighting Warrior» y nuestros aguerridos Justicieros del Software vienen dispuestos a todo.
Esta es su desinteresada y objetiva opinión.







JAVIER REDONDO. Madrid

La pantalla, al aparecer escalonadamente, provoca que prestemos mucha atención.

Los gráficos están bien hechos, con detalles bastante buenos, por ejemplo, la musculatura de los personajes, está muy lograda lo que unido al juego de sombras, le confiere una mayor sensación de realidad.

Todas las posturas que toma el personaje, son bastante correctas pero en la ambientación del fondo, al ser un desierto, quizá hay demasiadas palmeras.

Los movimientos, teniendo en cuenta la cantidad de detalles que tienen los gráficos, son rápidos, aunque se echa de menos una variedad de golpes.

Un aspecto negativo es que al efectuar otro golpe o movimiento diferente es preciso que el personaje esté en su posición inicial y la transición del golpe al volver a su posición inicial es bastante lento.

El sonido deja bastante que desear al ver aparecer la bestia. Creo que debería oírse algo más variado.

GORKA POLITE. Pamplona

La carga de la cinta es impresionante. Nos encontramos ante un difícil y bello programa en la línea del Exploding Fist, por lo menos en cuanto a gráficos y estilo de programación.

Trata de un joven valeroso que intentará salvar a la princesa, sirviéndose de su agilidad y de su espada que podrá utilizar en tres golpes diferentes.

Los gráficos son bastante buenos y están acompañados de movimiento, aunque quizá algo lento. El sonido es escaso, pero de buena calidad.

La dificultad y adicción son muy elevadas.

La pega del programa es que existen pocos golpes y las luchas se realizan muy de cerca por lo que no es nada fácil esquivar los golpes.

En resumen, es un bonito juego, muy difícil y eso sí, con mucha adicción, recomendado, sobre todo, a personas con mucha paciencia.

DANIEL GONZALEZ. Madrid

Está muy bien, lo malo es que hay pocos escenarios diferentes.

El movimiento está muy logrado y los gráficos son muy buenos.

Está bien que haya más de un contrincante, por lo que le da más emoción. La pena es que hay pocos movimientos de ataque.

Una vez que se llega al final y se rescata a la princesa, ocurre una cosa muy simpática que no cuento para que no pierda interés.

El sonido está bien.

En cuanto a originalidad es bastante parecido al Exploding Fist pero mejor realizados los personajes.

Es bastante complicado y la acción es rápida, aunque los personajes van un poco despacio. Yo lo calificaría de divertido y emocionante.



CELIA MENENDEZ. Madrid

Bastante parecido al Exploding Fist en cuanto a movimientos, gráficos y lucha en general. Las peleas, sin embargo, un poco lentas.

Cuando andamos hacia adelante el scroll de la pantalla es muy suave y está bien logrado. En algunos casos no tiene suficiente rapidez de reflejos, ya que por ejemplo, si estamos a mitad de un golpe y nos disparan una flecha, nos nos podemos agachar hasta que acabe el golpe.

El sonido no es muy positivo, ya que sólo se oye el ruido de la espada cuando luchamos.

La aparición de los objetos hace más ameno el juego ya que influye mucho la utilización de los mismos.

Lo mejor del juego son los gráficos.

El número de vidas en este caso, es adecuado.



JOSE C. NUÑEZ. Santander

Es un juego de gráficos bastante aceptable, aunque no haya muchos.

La calidad es excelente. Los movimientos del personaje, son fáciles de manejar con un poco de práctica.

En cuanto al sonido, no se han esforzado mucho.

El nivel de dificultad es alto, aunque también entra en juego la suerte, lo cual hace mucho más emocionante el juego.

Está bien ambientado, en relación a la historia, aunque quizá un defecto es que puede llegar a ser un poco monótono.



PURIFICACION HERNANDEZ. Burgos

Los gráficos están muy bien logrados y los personajes se mueven con mucha naturalidad y realismo.

Hay un inconveniente: que no se puede rectificar la marcha del personaje y hay que realizar toda la acción para cambiar a otro movimiento.

Tiene pocas modalidades de golpes (sólo tres), aunque ello no resta vistosidad al juego.

El sonido no es demasiado bueno, pero suficiente para este tipo de programas.

Es muy adictivo y su grado de dificultad es bastante alto.



M.ª A. GAGO. Cádiz

Comienza con una rutina de carga bastante espectacular, en donde se mezclan colores muy llamativos con una gran rapidez y vistosidad en los rótulos de los gráficos.

Una vez metidos de lleno en el juego, el decorado está bastante bien logrado. Sin embargo, se echa de menos una mayor capacidad de respuesta a la hora de manejar nuestro guerrero.

Algunos movimientos se ejecutan sólo después de la terminación de un movimiento de nuestro rival, lo que no es nada agradable.

El arte de la esquiva brilla aquí por su ausencia y hemos de limitarnos a un intercambio de golpes del que tan sólo, esporádicamente, saldremos vencedores.



G. VEGA. Madrid

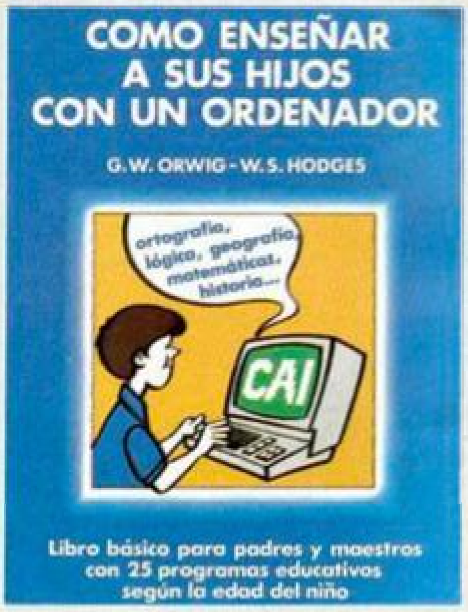
Es un juego muy entretenido con unos gráficos bastante buenos aunque no son muy variados.

Un defecto importante es la lentitud de movimientos.

En cuanto al sonido, tampoco tiene mucha variación aunque resulta original escuchar el ruido de las espadas cuando pelean.

La historia es entretenida y el juego, en definitiva, resulta difícil de ejecutar.

LIBROS



COMO ENSEÑAR A SUS HIJOS CON UN ORDENADOR

G. W. Orwing/W. S. Hodges

Ed. Gustavo Gili. 208 páginas

Este libro está enfocado al aprendizaje de los niños basado en ordenadores personales y será de una inestimable ayuda para todos aquellos padres o educadores que están convencidos de que un ordenador sirve para bastante más cosas que jugar y están decididos a demostrárselo a su hijos.

Se trata de una completa recopilación de programas elementales con sus correspondientes comentarios para el educador escritos todos ellos en BASIC estándar. Cada una de las particularidades de cada ordenador en concreto deberán ser adaptadas por el propio usuario con unos mínimos conocimientos de programación.

Los apartados están comentados por un personaje ficticio llamado Arturo y todos los programas están acompañados de unas notas al margen que advierten de las posibles modificaciones o variantes.

Los programas son bastante simples y utilizan técnicas de programación sencillas ya que se pretende que la persona que los teclee sea capaz de comprenderlos, modificarlos y adaptarlos a cada uno de los ordenadores en concreto.

Problemas de matemáticas, test de memoria, ejercicios de lenguaje, enigmas, matemáticas y física son algunos de los diversos temas que abarca este compendio de programas.

Incluye también un completo glosario de términos informáticos así como una bibliografía complementaria (es una lástima que todos los títulos sean en Inglés).

MICRO Manía

ROBIN OF SHERWOOD

Si os encontráis con dificultades para salir de la mazmorra donde Robin se encuentra prisionero al principio, probad con los siguientes pasos (pero en inglés, por supuesto).

– Súbete a los hombros de otro prisionero.

– Espera a que se oigan los pasos del guarda.

– Agárrale por el tobillo.

– Estrangúlale.

– Regístrale (verás que tiene una espada).

– Coge la espada.

– Rompe el candado.

– Abre la rejilla.

Y así conseguirás escapar y podrás continuar con la aventura.

NIGHTSHADE

Para destruir a los diferentes enemigos deberás usar para cada uno de ellos la siguientes armas:

Biblia para el Fantasma.
Mazo para el Esqueleto.
Reloj para la Muerte.
Cruz para el brujo.

EL HOBBIT

Para alcanzar el anillo en este juego, realiza los siguientes movimientos:

SE N SE U W SE S N SW

Una vez conseguido el anillo, si te diriges hacia:

NE NW E

llegarás hasta la puerta trasera.

JET PAC

Te ofrecemos un interesantísimo Poke gracias al cual podrás disfrutar de un número infinito de vidas, toma bien nota de él:

POKE 36966,224

MICRO HIT

1. **DUN DARACH** *(Gargoyle Games)*
2. **JET SET WILLY 2** *(Software Proyects)*
3. **ROCKY HORROR SHOW** *(CRL)*
4. **EXPLODING FIST** *(Melbourne House)*
5. **HYPERSPORTS** *(Imagine)*
6. **SPY HUNTER** *(U.S. Gold)*
7. **FRANKIE GOES TO HOLLYWOOD** *(Ocean)*
8. **DAM BUSTERS** *(U.S. Gold)*
9. **WORLD SERIES BASKETBALL** *(Elite)*
10. **HERBERT'S DUMMY RUN** *(Mikro-Gen)*
11. **SPY V SPY** *(Beyond)*
12. **CAULDRON** *(Palace)*
13. **NIGHTSHADE** *(Ultimate)*
14. **EVERYONE'S A WALLY** *(Mikro-Gen)*
15. **NODES OF YESOD** *(Odin)*
16. **PROFANATION** *(Dinamic)*
17. **BEISBALL** *(Imagine)*
18. **HIGHWAY ENCOUNTER** *(Vortex)*
19. **GYRON** *(Firebird)*
20. **NIGHTLORE** *(Ultimate)*



COMPUTIQUE

Te da más

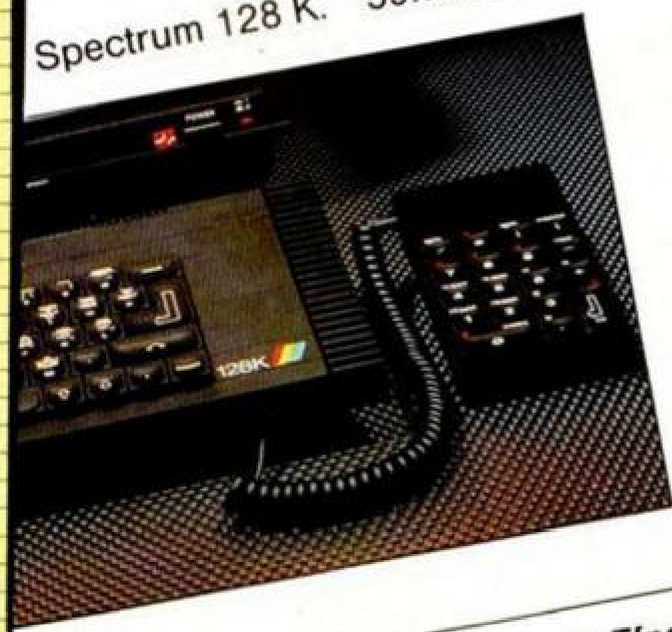
GARANTIA investronica



VENTAS A PLAZOS

¡NOVEDAD!

Spectrum 128 K. 59.700 ptas.



Al comprar tu spectrum *te regalamos*

- Joystick Gunshot I
- Estuche con ocho juegos originales
  - Psst
  - Chequered Flag
  - Flight Simulation
  - Cookie
  - Chess
  - Jet Pac
  - Reversi
  - Backgammon
- Un estupendo libro de Basic
- Los cuatro mejores Hits
  - Decathlon
  - Jet Set Willy
  - Sabre Wulf
  - Beach Head
- Y además te obsequiamos con un curso de introducción al Basic en nuestras aulas

Embajadores, 90 Tfno. 2270980
28012 Madrid

Servimos a tiendas
Abrimos sábados por la tarde

COMPUTIQUE

Profesor particular



Arturo LOBO y Juan J. LEON

RESOLUCION DE TRIANGULOS

El programa de esta semana está dedicado a una parte del temario de Matemáticas de los cursos 2.° y 3.° de B.U.P., y aunque va destinado principalmente a los alumnos de estos cursos, será de gran utilidad para todos los lectores que tengan dificultades con la trigonometria en general y los problemas de triángulos en particular.

El programa, además de ser didáctico, pretende ser ameno y exige vuestra colaboración para resolver el problema particular que le planteéis.

Llega a la solución siguiendo las mismas fases que seguiria una persona para solucionar el problema; además, muestra cómo solventar cada una de las fases y pregunta si está entendida, no pasando a la siguiente hasta que se haya comprendido la que está en pantalla.

Muestra en la pantalla un triángulo general con todas sus medidas y ángulos, asi como las fórmulas que va a utilizar para resolver el problema. Tras esto, pide los datos del problema y la incógnita que se desea hallar.

Para hacerlo mira si puede despejar directamente la incógnita de alguna de las fórmulas; si no es así lo indica y halla alguna otra incógnita con la cual podria dar con la que se ha pedido.

El problema debe estar bien planteado, pues en caso contrario halla una solución falsa o bien no obtiene ninguna. Un caso que sirve ahora de ejemplo es introducir como datos que los catetos tienen una longitud de 5 unidades y la hipotenusa mide 3 unidades.

Para detener el programa basta con pulsar ENTER cuando pida la incógnita.

Es necesario que los ángulos se introduzcan en radianes. Para ello recordad que 60°= PI/3, 30°=PI/6, 45°=PI/4; en general:

$$x\ (\text{rad.}) = \frac{x^{\circ} \cdot PI}{180^{\circ}}$$

Ejemplo: Datos α=PI/3 y ENTER, ß=PI/6 y ENTER, d=5 y ENTER; INCOGNITA: b y ENTER. Solución: b=2,5.

No olvidéis que siempre ha de darse $\alpha<\beta$ y $a>b$ y $c>d>h$; ha de tenerse muy en cuenta a la hora de introducir los datos.

TODAS LAS LETRAS MAYUSCULAS SUBRAYADAS DEBERAN TECLEARSE EN MODO GRAFICO

```
  10 BORDER 6: PAPER 6: INK 0
  11 CLS
  15 DIM Z$(11,3)
  16 RESTORE 211: FOR i=1 TO 10: READ Z$(i): NEXT i
  20 RESTORE 30: FOR i=USR "a" TO USR "a"+39: READ u: POKE i,u: NEXT i
  30 DATA 0,0,59,68,68,76,55,0
  31 DATA 0,16,16,56,40,68,68,64
  35 DATA 0,4,104,16,40,68,68,56
  40 DATA 0,12,16,16,8,52,36,24
  50 DATA 96,144,32,64,240,0,0,0
 100 PRINT AT 10,24;"a": PRINT AT 9,20;"h": PRINT AT 9,27;"b": PRINT AT 9,29;"b": PRINT AT 5,31;"h": PRINT AT 6,27;"d": PRINT AT 4,24;"c": PRINT AT 2,29;"c": PRINT AT 3,30;"c"
 110 PLOT 143,96: DRAW 105,0: DRAW 0,71: DRAW -105,-71: PLOT 207,96: DRAW 41,71: PLOT 143,87: DRAW 105,0: PLOT 140,84: DRAW 3,3: PLOT 244,84: DRAW 3,3
 120 PLOT 152,96: DRAW -2,6,0.63: PLOT 216,96: DRAW -4,7,1.1: PLOT 248,155: DRAW -7,4,-0.24: PLOT 248,151: DRAW -8,8,-0.4
 125 FOR i=0 TO 15: PRINT PAPER 6;AT i,0;"": NEXT i
 130 FOR i=1 TO 10: READ E$: PRINT PAPER 6; INK 7;AT 2*i-2,0;E$: NEXT i
 140 DATA "C=PI/2-A","SIN A=h/c","COS A=a/c","TAN A=h/a","S=ah/2","C=PI/2-B","SIN B=h/d","COS B=b/d","TAN B=h/b","S=bh/2"
 145 RESTORE 150: LET B$="12345 6789": LET raro=VAL B$: FOR i=1 TO 9: READ I$: GO SUB 9999: NEXT i: LET i=0
 150 DATA "a","b","c","d","h","u","v","w"
 155 BEEP 0.5,10: PRINT #1; PAPER 1; INK 6;"A,B,C,E estan en a,b,c,d en modo GRAFICO,respectivamente.": BEEP 2,20: PAUSE 150
 160 INPUT "METE LOS DATOS ";LINE L$;"";LINE M$;"";LINE N$
 165 LET linea=12
 170 INK 1: PRINT AT 0,15;"DATOS": PRINT AT 1,15;L$: PRINT AT 2,15;M$: PRINT AT 3,15;N$: INK 0
 179 LET pos=0: BEEP .5,10: PRINT #1; PAPER 1; INK 6;"RECUERDA CUANDO TE PREGUNTE ""ENTENDIDO"" PULSA UNA TECLA": BEEP 2,20: PAUSE 100
 180 INPUT "CUAL ES LA INCOGNITA ? ";LINE V$: PRINT AT linea,14;"": LET pos=pos+1
 181 IF V$="" THEN STOP
 182 LET datos=3: IF N$="" THEN LET datos=2
 185 RESTORE 186: FOR z=1 TO datos: READ e$: GO SUB 1500: GO SUB 8980: GO SUB 9999: NEXT z
 186 DATA l$,m$,n$
 188 LET e$=V$: GO SUB 1500: LET V$=e$
 189 LET x=raro+2*((V$="v")+(V$="w"))
 190 DEF FN d$(i$)=CHR$ ((CODE i$-28)*(CODE i$>143)*(CODE i$<148)+(CODE i$+28)*(CODE i$>115)*(CODE i$<120)+CODE i$*((CODE i$<116)+(CODE i$>119))*((CODE i$<144)+(CODE i$>147)))
 192 FOR o=1 TO datos: IF pos=1 THEN IF ((Z$(11,o)="v")+(Z$(11,o)="w"))=1 THEN LET x$=CHR$ ((Z$(11,o)="v")*117+116*(Z$(11,o)="w")): LET i2=i: LET i=2: GO SUB 500: LET i=i2
 194 NEXT o
 201 DATA "w=PI/2-v"
 202 DATA "SIN t=h/c"
 203 DATA "COS t=a/c"
 204 DATA "TAN t=h/a"
 205 DATA "a†2+h†2=c†2"
 206 DATA "v=PI/2-u"
 207 DATA "SIN v=h/d"
 208 DATA "COS v=b/d"
 209 DATA "TAN v=h/b"
 210 DATA "b†2+h†2=d†2"
 211 DATA "wtt","thc","tac","tha","ahc","vut","vhd","vbd","vhb","bhd"
 300 LET i$=v$: GO SUB 500: BEEP .5,25: PRINT INVERSE 1; FLASH 1;AT linea,14;"FIN DEL PROBLEMA": GO TO 180
 500 IF VAL v$<>raro THEN RETURN
 505 LET p3=0: LET p2=0: FOR z=1 TO 10: FOR m=1 TO 3: IF Z$(z,m)=i$ THEN GO TO 520
 510 NEXT m: NEXT z: GO TO 600
 520 LET itas=0: FOR m=1 TO 3: IF VAL Z$(z,m)=raro THEN LET itas=itas+1
 525 NEXT m: IF itas=1 THEN LET i$=i$: RESTORE (200+z): READ e$: GO SUB 9000: GO SUB 9999: LET i$=i$+"="+b$: GO SUB 1500: PRINT AT 20,1;"DE (";z;") TENGO ";E$: GO SUB 2000: BEEP .5,50: PRINT AT linea,13;"": PRINT INK 2;AT linea,16;FN d$(i$);"=";VAL i$: LET linea=linea+1: RETURN
 540 IF itas=2 THEN LET p2=z: NEXT z: GO TO 600
 545 LET p3=z: NEXT z: GO TO 600
 600 GO SUB 700: PRINT INK 2;AT 20,0;"NO PUEDO DESPEJAR ";FN d$(i$);" DIRECTAMENTE": GO SUB 2000: PRINT AT 20,0;"Hallo primero ";FN d$(Z$(p2+p3*(p2=0),m));" para obtener ";FN d$(i$);" de (";p2+p3*(p2=0);")": GO SUB 2000: LET i$=Z$(p2+p3*(p2=0),m): GO SUB 500: LET i$=v$: GO TO 500
 700 FOR m=1 TO 3: IF (Z$(p2+p3*(p2=0),m)<>v$)*(Z$(p2+p3*(p2=0),m)<>i$)*(VAL Z$(p2+p3*(p2=0),m)=raro)=1 THEN RETURN
 710 NEXT m: STOP
1500 LET s$=FN d$(e$(1)): FOR i=2 TO LEN e$: LET s$=s$+FN d$(e$(i)): NEXT i: LET e$=s$: RETURN
2000 BEEP .5,10: PRINT AT 21,20;"ENTENDIDO ?": PAUSE 0: PRINT AT 20,0;"": RETURN
8980 FOR i=1 TO LEN e$: IF FN g(e$(i))=1 THEN LET i$=e$(i): LET Z$(11,z)=i$: GO TO 9000
8985 NEXT i: BEEP 2,10: PRINT "EXPRESION NO VALIDA": RUN
8990 DEF FN g(i$)=(CODE i$>64)*(CODE i$<91)+(CODE i$>96)*(CODE i$<123)
9000 FOR i=1 TO LEN E$: IF E$(i TO i+LEN I$-1)=I$ THEN LET p=i: GO TO 9010
9005 NEXT i
9010 FOR i=1 TO LEN E$: IF E$(i)="=" THEN LET pe=i: GO TO 9020
9015 NEXT i: PRINT "NO HAY ""=""": RETURN
9020 IF pe<p THEN LET B$=E$(1 TO pe-1): LET A$=E$(pe+1 TO LEN E$): GO TO 9030
9025 LET B$=E$(1 TO (pe-1)): LET A$=E$((pe+1) TO LEN E$)
9030 FOR i=1 TO LEN A$: IF A$(i TO i+LEN I$-1)=I$ THEN LET pi=i: GO TO 9035
9032 NEXT i
9035 IF A$=I$ THEN RETURN
9040 LET cuenta=0: DIM a(25,2): DIM b(25): LET a=1: LET np=0: LET a(1,2)=0: FOR i=1 TO LEN A$: IF CODE A$(i)=40 THEN LET cuenta=cuenta+1: LET np=np+1: IF np=1 THEN LET a=a+1: LET a(a,1)=i
     NEXT i
9045 IF CODE A$(i)=41 THEN LET np=np-1: LET cuenta=cuenta-1: IF np=0 THEN LET a(a,2)=i: NEXT i
9047 NEXT i: IF cuenta<>0 THEN PRINT "numero impar de parentesis": RETURN
9050 LET a(a+1,1)=LEN A$+1: IF a(2,1)=1 THEN IF a(2,2)=LEN A$ THEN LET A$=A$(2 TO (LEN A$-1)): LET pi=pi-1: IF a(a,2)<>0 THEN GO TO 9035
9055 LET no=0: LET niv=0: LET b(1)=0: FOR j=1 TO a: FOR i=a(j,2)+1 TO a(j+1,1)-1: LET orden=FN p(CODE A$(i)): IF orden>niv THEN LET niv=orden: LET no=2: LET b(2)=i: GO TO 9062
9060 IF orden=niv THEN IF niv<>0 THEN LET no=no+1: LET b(no)=i
9062 NEXT i: NEXT j: LET b(no+1)=LEN A$+1
9070 GO SUB (9100+100*niv): GO TO 9030
9080 DEF FN p(i)=3*((i=43)+(i=45))+2*((i=42)+(i=47))+(i=94)
9100 IF CODE A$(1)=187 THEN LET A$=A$(2 TO LEN A$): LET B$="ABS ("+B$+")": RETURN
9105 DEF FN I(i)=189*(i=189)+45*(i=47)+43*(i=45)+179*(i=181)+181*(i=179)+179*(i=182)+182*(i=179)+180*(i=183)+183*(i=180)+184*(i=185)+185*(i=184)+42*(i=47)+47*(i=42)
9110 LET B$=CHR$ FN I(CODE A$(1))+"("+B$+")": LET A$=A$(2 TO LEN A$)
9115 IF FN I(CODE A$(1))<>0 THEN LET B$=CHR$ FN I(CODE A$(1))+B$: LET A$=A$(2 TO LEN A$): GO TO 9115
9120 RETURN
9200 FOR i=1 TO no: IF b(i)<pi THEN LET pos=i
9205 NEXT i
9210 IF pos=1 THEN LET C$=A$(b(2)+1 TO b(3)-1): FOR i=3 TO no: LET C$=C$+"+"+A$(b(i) TO b(i+1)-1): NEXT i: LET B$="ABS ("+B$+")†(1/("+C$+"))": LET A$=A$(1 TO b(2)-1): RETURN
9220 LET C$="ABS "+A$(1 TO b(2)-1): FOR i=2 TO no: IF i<>pos THEN LET C$=C$+"*"+A$(b(i)+1 TO b(i+1)-1)
9230 NEXT i: LET B$="LN ABS ("+B$+")/LN ABS ("+C$+")": LET A$=A$(b(pos)+1 TO b(pos+1)-1): RETURN
9300 GO TO 9400
9400 LET B$="("+B$+")": IF b(2)<>1 THEN LET A$=CHR$ (42+(niv=3))+A$: FOR i=1 TO no+1: LET b(i)=b(i)+1: NEXT i: LET pi=pi+1: GO TO 9410
9405 FOR i=1 TO no: LET b(i)=b(i+1): NEXT i: LET no=no-1
9410 FOR i=1 TO no: IF b(i)<pi THEN LET posx=i
9415 NEXT i
9420 IF A$(b(posx))=CHR$ (42+(niv=3)) THEN GO TO 9440
9425 LET C$="": FOR i=1 TO no: IF i<>posx THEN LET C$=C$+A$(b(i) TO b(i+1)-1)
9430 NEXT i: LET C$=C$(2 TO LEN C$): LET B$=C$+CHR$ (45+2*(niv=2))+B$: LET A$=A$(b(posx)+1 TO b(posx+1)-1): RETURN
9440 FOR i=1 TO no: IF i<>posx THEN LET B$=B$+CHR$ FN I(CODE A$(b(i)))+A$(b(i)+1 TO b(i+1)-1)
9445 NEXT i: LET A$=A$(b(posx) TO b(posx+1)-1): RETURN
9980 STOP
9999 LET a(0)=PEEK 23627+256*PEEK 23628: POKE (a(0)-8),CODE i$: LET d=VAL b$: RETURN
```

CONSULTORIO

Overflow

En pocas palabras mi duda es la siguiente: ¿Cómo se distingue en Código Máquina si un número es positivo o negativo? Debido a la representación de los números negativos por el complemento ocurre que los números del −1 al −225 equivalen correlativamente a los del 255 al 1, por lo que son indiscernibles, entonces, ¿cómo se puede reconocer que el contenido de una posición de memoria que se toma como 225 es realmente 225 y no −1?

Este problema me surgió al tener que escribir una rutina en Código Máquina que controlase los resultados de la diferencia entre dos números, de forma que éstos se encuentren en el rango de 127 a −128 (precisamente el de los números con signo en un solo octeto) pero el resultado de la diferencia puede muy bien ser desde −129 a −225 ó de 128 a 225, números que se confunden con los anteriores y que de alguna forma he de identificar para rechazarlos como resultados erróneos. ¿Cómo se resuelve esta cuestión?

Francisco BUENO - Madrid

☐ Tras una operación aritmética se activan una serie de "Flags" o indicadores en el registro "F" del microprocesador. Uno de estos indicadores es el "P/V" que en las operaciones aritméticas actúa como indicador de "OVERFLOW" (Rebosamiento) e indica que el resultado ha rebasado el margen permitido.

En nuestro curso de Código Máquina se tratará esta cuestión con mayor amplitud.

Over 1

¿Cómo se podría conseguir que un dibujo hecho con las sentencias "PLOT" no se borrara al pasar un gráfico por encima de ellas?

¿Cómo se teclean los programas que no están en Basic, como los del número 20 en la página 27?

¿A qué dirección se deben mandar los programas? Si se publican, ¿mediante que sistema se adquiere el premio?

Arturo MARTINEZ - Madrid

☐ Para que un dibujo no se borre al pasar un gráfico por encima existen dos posibilidades: Una imprimir "OVER 1", y otra, guardar en algún sitio el contenido anterior de la cuadrícula antes de imprimir, para recuperarla después.

Los programas que no están en Basic son listados en Assembler. Puede introducirlos con ayuda de un programa «Ensamblador», pero no le será necesario, ya que siempre publicamos el correspondiente programa en Basic que se encarga de hacer la introducción. Los listados en Assembler se publican a título indicativo, para que el lector comprenda el funcionamiento de las rutinas, o por si quiere ensamblarlos en otra dirección.

Puede enviar sus programas a nuestra dirección en la calle La Graja. Si se publican, recibirá un talón nominativo sobre el que ya se habrá hecho la preceptiva retención del 5% por I.R.P.F.

Ensamblador

Desearía me dijeran el equivalente en lenguaje Ensamblador de esta instrucción Basic:

10 PRINT AT 5,12;"A"

¿Para qué sirve la instrucción: Ld A,A, y cómo se puede imprimir el contenido de un registro en el Ensamblador?

Abelardo ORTIZ - Valencia

☐ No es posible establecer una equivalencia directa entre instrucciones en Basic y en Assembler. No obstante, si lo que quiere es imprimir en 5,12 el contenido del acumulador, puede utilizar la rutina RST 16 de la ROM.

Otra forma posible de sacar datos a pantalla es cargarlos en "BC" antes de retornar, y llamar a la rutina con "PRINT USR ... "

Respecto a la instrucción LD A,A no sirve realmente para nada, es más bien una consecuecia del formato de instrucción que utiliza el Z-80 para cargar registros.

Joystick

¿Por qué no listáis vuestros programas para que puedan ser usados con Joystick?

En los demás programas, ¿hay alguna forma de adaptarlos para Joystick?

Hilario GUIJO - Madrid

☐ La razón de que nuestros programas no se publiquen adaptados para Joystick, es que no todos nuestros lectores disponen de este periférico, y en caso de disponer de él, unos lo tienen según norma Kempston y otros, según norma Sinclair.

Por otro lado, la adaptación es muy sencilla, basta con encontrar las líneas que leen el teclado, y cambiarlas para que lean el Joystick. A este respecto, puede ser util una atenta lectura de los artículos aparecidos en los números 33 y 34 de nuestra revista.

DE OCASION

● VENDO impresora GP-50S por 20.000 ptas. y monitor «Fetaki» «fósforo verde» con interface para el monitor incluido por 25.000 ptas. También vendo un ZX Spectrum Plus por 30.000 ptas. Todo el lote completo con cables, cinta de demostración y más de 50 revistas por 70.000 ptas. Dirección: Avda. Alfonso Sala, 30,. Sant Cugat del Valles (Barcelona). Tel. (93)6742928 (preguntar por Daniel Trell.

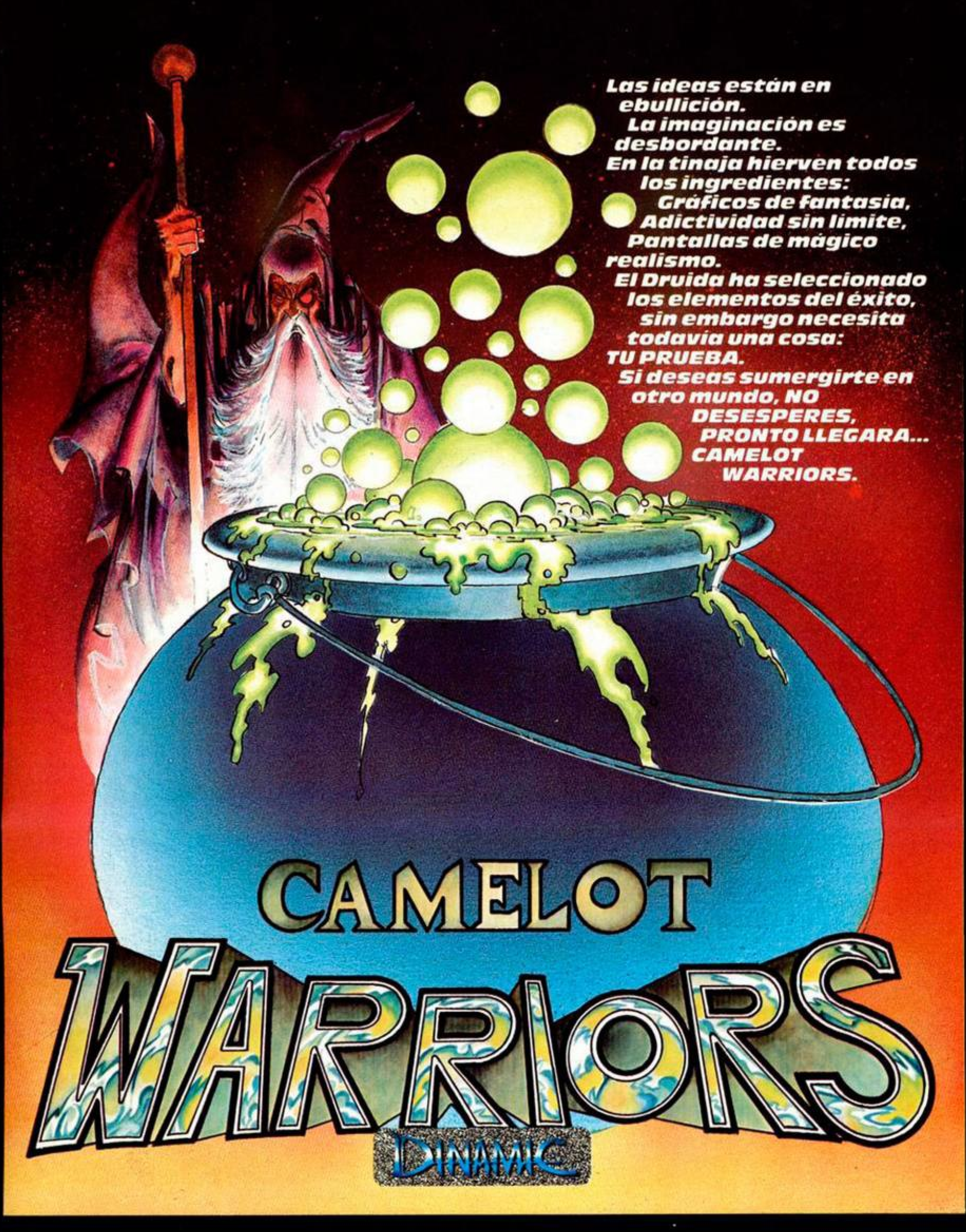
● DESEARIA contactar con usuarios del Spectrum de Sabadell, Barbera del Valles y Ciudad Badia para intercambio de ideas, información. Llmar al Tel. 7181774 (a partir de las 4 de la tarde). Preguntar por Eusebio.

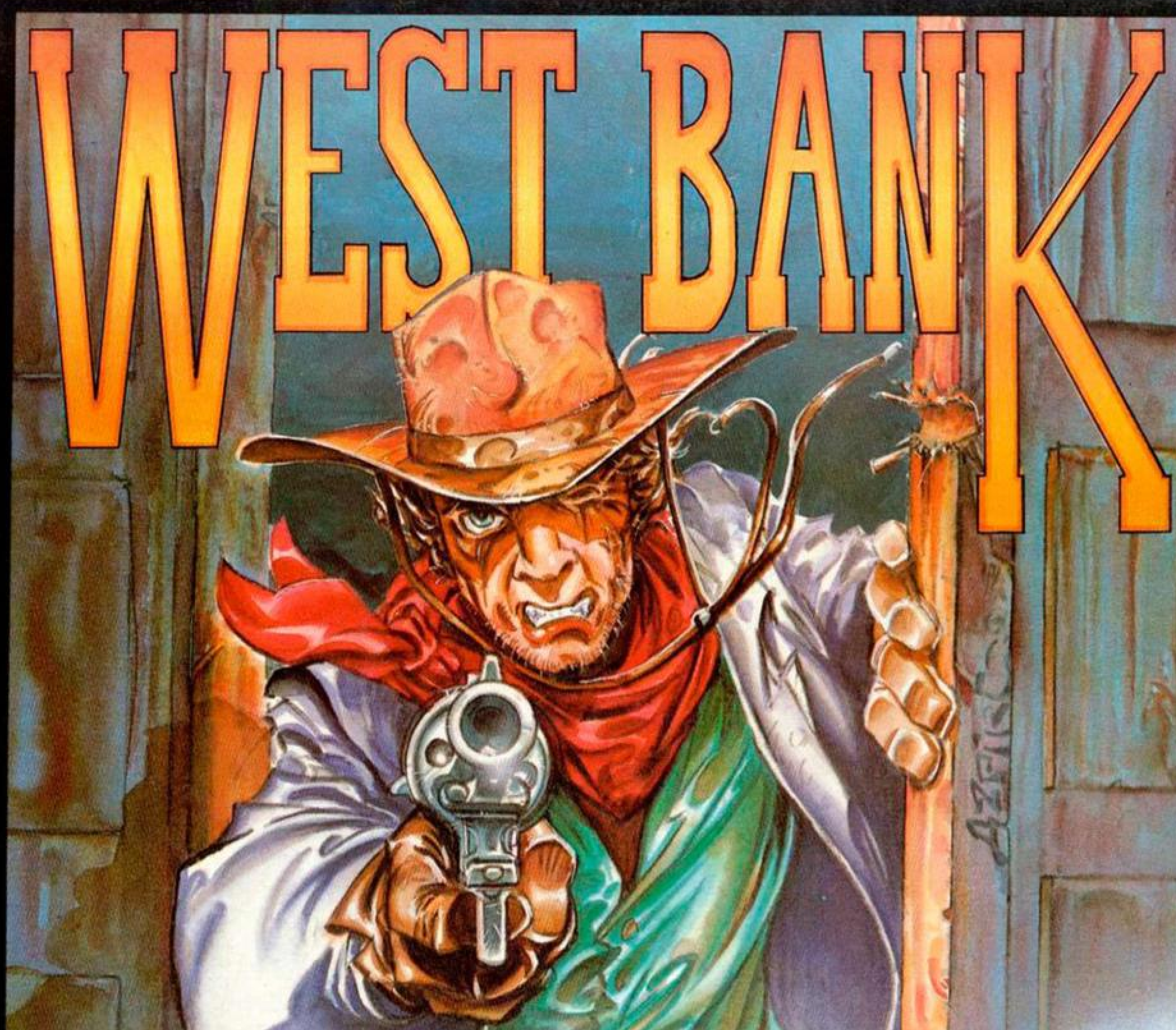
● VENDO Sinclair ZX Spectrum 48K, con cables, transformador y manual de instrucciones en castellano. Junto al ordenador también el monitor profesional especial para computadoras con todos sus accesorios. Todo por 40.000 ptas. Llamar o escribir a Rafael de Mingo Gil. Dinan, 13 G. Madrid. Tel. 7418859.

● VENDO Spectrum 48K, microdrive, interface 1 (garantía en blanco, manuales en español), más de 60 revistas, libro de Basic de «Microhobby». Todo por 59.000 ptas. También Casio PB-110 por 10.000 ptas. Interesados llamar al Tel. 7770167 de Madrid.

● VENDO Spectrum 16K, en perfecto estado, con manual de instrucciones. Precio: 15.000 ptas. Contactar con Javier Aldea. Paseo Valdaura, 237, 1.º Barcelona o llamar al Tel. 3502272.

¿QUE SE CUECE EN LA MANSION DINAMIC?





Fotos tomadas de un Spectrum 48 K

SPECTRUM · 1950 · AMSTRAD
(DISK · 3050)

PEDIDOS CONTRA REEMBOLSO:
TEL.: (91) 715 00 67
TIENDAS Y DISTRIBUIDORES:
TEL.: (91) 447 34 10.

DINAMIC

«MANSION DINAMIC»
c/ TILOS, 2, 21,
MONTEPRINCIPE
BOADILLA DEL MONTE
MADRID